階層的空間索引の効率的構築に関する研究

著者	今村　安伸
その他のタイトル	A Study on Efficient Construction of Hierarchical Spatial Indexes
学位授与年度	平成30年度
学位授与番号	17104甲情工第336号
URL	http://hdl.handle.net/10228/00007206

階層的空間索引の効率的構築に関する研究

今村 安伸

概要

画像や音などの高次元データを含むマルチメディアデータの検索においては，完全一致検索では有用な結果を得ることが難しく，類似検索の需要が高い．類似検索を素朴に処理すると毎回すべてのデータを走査する必要があり，データ量が増えると現実的ではなくなっていく．そこで，空間索引を用いた効率的な類似検索が研究されてきた．空間索引では，検索対象データを多次元空間の点とみなし，質問点の近傍点の検索を行う．本論文では，階層的空間索引の効率的構築のための手法について，木構造索引の構成時のクラスタリング手法であるヒルベルトソートおよび次元の呪いの緩和のための次元縮小法を取り上げて，それらの効率的解法を提案する．

空間索引の一つである R-tree は，近いもの同士を超矩形としてまとめてノードとして扱う階層的索引構造である．このときの超矩形は超立方体に近い方が望ましいとされている．ヒルベルト曲線では連続したいかなる区間でも，それを包括する超矩形は超立方体に近くなる．この特性を利用し，ヒルベルト曲線順を元にしたクラスタ分けによってノードを構築する Hilbert R-tree がある．ヒルベルト曲線の生成はデータの次元数および曲線の細かさを表す次数に対して指数オーダーのコストがかかるので，高次元高次数においてはヒルベルト曲線を生成せずにヒルベルト曲線順にデータ点を並べ替えることが望まれる．従来法として，データ点の座標からヒルベルト曲線順の番地を示すヒルベルトインデックスを求めて，インデックスをキーとして並べ替える方法があった．本論文では，ヒルベルトインデックスを求めずに，直接データ点をヒルベルト曲線順に並べ替える方法について議論した．直接ソートを行う際，複数の点に対してまとめてヒ

ルベルト曲線順をシミュレーションし，また，不必要な細部のシミュレーションを省略するため，大幅な速度改善が期待できる．また，実験によっても理論通りの速度上の優位性を確認した．また，提案手法を用いることにより，Hilbert R-tree から不要なヒルベルトインデックスを除去して省メモリ化と高速化を同時に達成することが可能となった．

検索対象となるデータ空間の次元が増えていくと，いかなる方法を用いても効率的な検索が難しくなっていくことが「次元の呪い」として知られている．次元の呪いに対する一つの緩和策として，次元縮小射影が存在する．次元縮小射影 Simple-Map では，空間内の点をピボットとし，ピボットとの距離を座標値として射影を行う．ピボット数だけ座標軸を得られ，ピボットを少なく抑えることで，低次元へと射影できる．次元縮小を行うと，距離の縮みが生じる．Simple-Map では，用いるピボット群によって，距離の縮み具合が変化する．距離情報をあまり失うことなく，つまり，距離の縮みを抑えて低次元へと射影できる良いピボット群を見つけ出す必要があり，これまでヒューリスティックな解法が研究されてきた．メタヒューリスティクスの一つである焼きなまし法（Simulated Annealing, SA）を用いた Simple-Map のピボット探索では，1 遷移あたりの評価コストが高いために遷移回数を増やすことができず，現実的な時間内ではヒューリスティックな解法より良い解を得ることができなかった．本論文では，増加再標本焼きなまし法（Annealing by Increasing Resampling, AIR）を提案する．AIR は，標本を用いる目的関数の最適化に適用することができる．通常の SA では，高温から低温に温度を変化させて最適化を行うが，AIR では，再標本の大きさを変化させる．目的関数を評価するたびに独立に選ぶ再標本を用い，小さな再標本で開始し，徐々に再標本を大きくしていく．AIR を用いることによって，SA の高温時に対応する 1 遷移あたりの評価コストを

桁違いに下げることに成功し，十分な遷移回数を確保することによってヒューリスティックな解法より良い解を得ることに成功し，階層的空間索引 R-tree を効率的に構築することを可能にした．

また，本論文では，増加再標本焼きなまし法 AIR について理論面において焼きなまし法 SA の近似になっていることを示した．元々の SA は，モンテカルロ法によるシミュレーションのメトロポリス法を起源とするものである．この状態遷移を行うかどうかの判断基準の受理条件は，視点を変えると，目的関数の評価値に確率的揺らぎをもったものを用いていると見ることができる．メトロポリス法での確率的揺らぎは，logit 関数で説明できる．この logit 関数を probit 関数へと置き換えるものとして AIR を定義することが可能である．一般に知られているように，logit 関数と probit 関数は近似できるため，AIR は SA の近似として機能することがわかる．本論文では，メトロポリス法において logit 関数を probit 関数に置き換えた場合にどのぐらいの遷移回数まで通常のメトロポリス法と同程度の精度を保てるのかについても実験による検証を行った．その結果，10^6 回程度以下での遷移回数であれば，両者の精度にほとんど違いがないことが確認でき，現実的な状況下においては AIR が SA の十分な近似であるといえる．さらに，SA の近似であることに着目をして，遷移回数などの条件をそろえて，従来の SA と提案手法の AIR との比較実験を行い，非常に近い性能の解が得られることを確認した．また，データ量増加に伴って，AIR に速度面で大幅な優位性が得られることも確認した．

目次

第1章　はじめに

この章では，論文の全体像について述べる．研究目的とその解決策である類似検索と空間索引を紹介する．空間索引でのクラスタリングや次元縮小法の必要性について述べ，また，焼きなまし法に代わる新しい手法として提案する増加再標本焼きなまし法の概要を紹介する．

1.1 類似検索と空間索引

近年，コンピュータの処理性能向上によりビッグデータなどデータ件数の巨大なデータを扱う時代へと変化してきている．完全一致検索では，データをバイナリ文字列として扱えば，辞書順に並んだデータの二分探索やハッシュなどの手法が使うことができ高速な検索が可能になる反面，内容を考慮せずに処理するため，ほんの少しでも異なるデータになるだけで検索することができなくなる．マルチメディアデータや観測データなどの自然データの検索においては，データベース側と検索要求側の両方のデータにノイズが含まれていることが多く，また，似た前例を探す要求なども多く，完全一致検索では役に立たない．こうした問題を解決する方法の一つが，データの内容に基づく尺度で似たものを探す**類似検索**[44] である．本論文では，距離に基づく類似検索を取り扱う．そこでは，オブジェクト間に距離を導入し，距離が近いものを「似ている」と考える．つまり，類似検索とは，質問として与えられたオブジェクトに近いものを求めることである．質問には範囲質問と近傍質問の 2 種類があ

る．範囲質問は，質問点から半径 r 内の点を求める．k 近傍質問は，質問点から近い順に k 個の点を求める．

近年注目されている AI 技術の Deep Learning[28] では，クラス分類などの問題に対して精度の高い結果が得られても，結果の理由を説明することが難しいとされている．これに対して，説明可能な人工知能 XAI（Explainable Artificial Intelligence）[10] への要求が高まっている．類似検索では近い具体的な対象が見つかる場合には，近傍点自体が説明になっていると考えられるので，たとえば，Deep Learning の出力層近くの中間層を特徴量として類似検索を行うことでその説明の手がかりを得ることができるとかもしれない．

しかし，類似検索を素朴に処理すると，すべてのデータを毎回走査する必要があり，データ量が膨大になると現実的ではなくなる．本論文では，類似検索を高速に実現するための研究分野の一つである，距離情報を用いた空間索引について，効率化と高速化を取り扱う．

1.2 空間索引とクラスタリング

空間索引では，画像や音声などのマルチメディアデータを高次元空間中の点として抽象化し，2 点間には距離を定義する．探したいデータを示す質問点の近くにあるデータベース上の点を見つけることで，類似検索を実現する．オブジェクト間の距離を求めるコストはオブジェクトの次元数に比例すると考えられるので，素朴な順次探索による類似検索は，データベースのオブジェクト数と次元数の積に比例するコストを必要とし，次元数やオブジェクト数が大きい場合には，大量の

質問を短時間で処理するのが難しくなる．この問題を克服するためのオブジェクト数の影響を小さくする手法には，近いもの同士は近いという距離の特性を用いて，距離の近いオブジェクトをクラスタとしてまとめるクラスタリングを用いる階層的索引構造の R-Tree[14] や M-Tree[6, 43] がある．

R-tree では近い点同士を座標軸に並行な超矩形としてまとめてノードとし，ノード同士を更に階層的にまとめる木構造のデータ構造として扱う．このときの各ノードは，質問範囲との交差確率が小さい方が枝刈り効率が良く，検索効率が良くなる．交差確率を小さくするには一見して超体積を小さくすることが良さそうだが，その最適形である短冊状のノードでは，特定の軸の長さは極端に小さいが他の軸は空間全体を覆うほどに大きくなりやすく，実際には交差確率はとても大きくなる．交差確率を小さくするには，ノード内のすべての点同士が互いに近くなるようにまとめるのが効果的である．なぜならば，質問点とノード内の 1 点の距離が遠ければ，ノード内のその他の点についてもやはり質問点と遠いことが確定するためである．これを満たすノードの超矩形は，超立方体に近くなる．マンハッタン距離である L_1 やユークリッド距離である L_2 の様な距離空間では，次元が増えれば増えるほど超球と超立方体との乖離が大きくなり，超矩形ノード内の角の点と角の点の距離が離れるため，交差確率の増加につながる．これに対して，L_∞ 距離空間では，超球の形は，通常の超立方体に相当するので，この乖離が存在しない．

いくつかの空間充填曲線では，連続した区間は実際の距離も近く，ヒルベルト曲線 [21] において特にこの特性は強い[12]．また，ヒルベルト曲線は任意次元数・任意次数へと適用可能である（図 1）．したがって，ヒルベルト曲線順に並んだデータは，ど

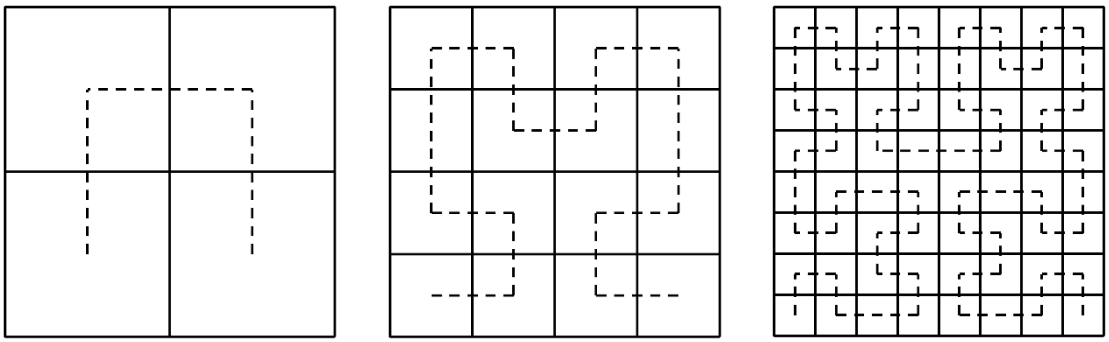


図 1　2 次元空間における次数 1, 2, 3 のヒルベルト曲線

の区間で分割しても，各区間は超立方体に比較的近くなる．この特性を利用した R-tree の亜種として，ヒルベルト曲線順を元にクラスタ分けを行う Hilbert R-tree[26] がある．ヒルベルト曲線の生成 [25] は空間の次元数および曲線の細かさを表す次数に対して指数のオーダーを持つため，高次元においてはヒルベルト曲線を生成することなくヒルベルト曲線順にデータ点をソートする必要がある．既知の改善された方法 [3, 15, 16] では，次元数および次数に対して線形時間で一つのヒルベルトインデックスを計算する．ヒルベルトインデックスの表現には座標点データと同じビット数を必要とし，必要なメモリ量はデータの 2 倍となる．しかし，データ点のソート後にはヒルベルトインデックスは不要である．

本論文では，ヒルベルトインデックスを求めずに，直接データ点をヒルベルト曲線順にソートする効率的な方法について議論する．この方法は，田中 [39] が導入したものである．この方法は，実際のマルチメディアデータでも人工的な一様乱数データでも，従来のヒルベルトインデックスを求める方法より高速に動作する．また，ヒルベルトインデックスを記録する領域を必要としないため，ソート処理だけでなく，階層的索引構造の省メモリ化を可能とする．

1.3 次元縮小法の必要性

検索対象となるデータ空間の次元が高くなると，効率的に検索することが難しくなっていくことは良く知られている．これは**次元の呪い**，つまり，高次元中ではそもそも近い点が存在する確率が非常に小さく，近いもの同士を集めることが難しくなることが原因の一つであると考えられている．ユークリッド距離空間や L_1 距離空間，文字列の編集距離空間 [41] などの L_∞距離空間以外の空間上の R-tree では，索引のためのクラスタリングで用いられる超矩形と，質問に用いられる超球の形が，次元が高いほど乖離が大きくなることも問題の一つである．次元の呪いに対する緩和策の一つが**次元縮小射影**である．

次元縮小射影では，高い次元から低い次元へと射影を行う．この射影後の空間での距離は，元の空間での距離を完全には保存しない．パターン認識 [13] の分野では，情報の整理されていない高次元データを，距離情報の劣化を抑えつつ情報の整理された低次元データへと射影する特徴抽出が次元縮小射影に相当している．伝統的な手法の多くは，主成分分析法（あるいはK-L 変換）やFastMap[11] のようにユークリッド距離を前提としている．これに対して，H-Map[37] や Simple-Map[38]（以下S-Map）は三角不等式を満たす任意の距離空間に適用できる次元縮小射影である．

S-Map では，ピボット [5, 29, 30] との距離を座標値として射影を行うもので，ピボットの数だけ座標軸を得ることができる．少ないピボットを用いて射影することで，低次元へと射影する．射影後の空間では L_∞距離空間として扱う．L_∞距離空間では質問に用いられる超球の形と，索引で用いられる超矩形の形が近くなるというメリットもある．S-Map では，距離が縮むことはあっても，伸びることがないという性質がある．これによ

って，S-Map 射影空間上で空間索引を構築すると，探したいデータを漏れることなく厳密解として得ることができる．ただし，遠いデータも近いと見なされてしまうため，本来不要なデータも多く検索対象となってしまい，これを除去する手間が検索時の性能低下へと結びつく．しかしながら，高次元上で検索することによる性能低下と比較すると，距離が縮むことによる性能低下の方が小さくて済み，高速な検索が可能となっている．これは乱数的なデータを除いて，ほとんどのデータには何らかの偏りがあり，本質的な次元は低くなっており，ある程度次元縮小を用いて次元数を下げても，あまり距離情報が失われないためであると考えられる．

類似検索で用いられる手法に局所性鋭敏ハッシュ（Locality Sensitive Hash, LSH）がある．**スケッチ** [9, 19, 20, 34, 35, 42] は，LSH の一種であり，オブジェクトをコンパクトなビット列で表現したもので，スケッチを用いる従来の検索ではスケッチ間のハミング距離を用いている．スケッチへの写像は，ハミング距離を用いている限り，次元縮小とはいえないが，質問とスケッチ間の距離下限を求める手法を用いれば，球面分割（BP）によるスケッチを次元縮小とみなすことができる [19]．BP とは，基準オブジェクトを中心とする球の内部と外部にそれぞれ 0 と 1 を割り当てるものである．BP を用いるスケッチは，S-Map のピボットからの距離を球の半径に相当する閾値以上であるかどうかに応じて 0 または 1 に量子化したもの [36] と考えることができる．

S-Map や BP で射影を行う際，用いるピボット群によって，距離の縮み具合や検索効率は変化する．良いピボット群では，もともとの空間で持っていた距離情報をうまく整理して，その距離情報をあまり失うことなく低次元へと射影することができるが，悪いピボ

ット群では，大量の距離情報を失うことになる．良いピボット群を用いることができるかどうかが，検索効率へと大きな影響をもたらす．S-Map のためのピボット群選択については，主成分分析法のように解析的手法が存在しない．このような組み合わせ最適化問題では，実際に射影した際の性能を測る以外に適切な評価指標が存在しない場合が多い．そうした場合には，解評価に射影対象のデータ標本を用い，個々のデータ評価の平均（あるいは合計）として標本全体を評価して，最適化を行う．このとき標本サイズが増えるほど，解評価のコストが大きくなり，効率的な探索が困難となる．

S-Map や BP スケッチのピボット群探索には，これまで，乱択や局所探索，焼きなまし法といった一般的な探索法や，データの分布特性を用いた二値量子化法（BQ）[18] などが用いられてきた．これらは，いずれも，標本を用いた評価値を指標とした最適化を行っている．S-Map では，距離保存率を評価値として，それを最大化する．BP では，衝突確率を評価値として，それを最小化する．本論文では，最適化手法として，**増加増再標本焼きなまし法**（Annealing by Incremental Resampling, AIR）という新たな方法を提案し，ピボット群探索での有効性を検証する．

1.4 増加再標本焼きなまし法の提案

焼きなまし法[27]（Simulated Annealing, SA）は，与えられた目的関数の大域最適値を探索する最適化問題を解くための確率的手法である．焼きなまし法は温度のパラメータを用いる．最初に，焼きなまし法は高温で始まる．このとき，探索空間内のとても広い範囲を探索対象としている．つづいて，温度をゆっくりと下げることによって，探索

範囲を徐々に狭めていく. 最終的には, 局所探索と同様にふるまう. S-Map のためのピボット探索問題においては, SA では, 現在の解候補からつぎの解候補への遷移あたりのコストが高いために, 現実的な探索時間内では遷移回数を増やすことができず, 軸別量子化法 [18] より良い解を得ることができなかった.

本論文では, **増加再標本焼きなまし法**（Annealing by Increasing Resampling, AIR）を提案する. AIR では, 評価に標本を用いる最適化問題で, 目的関数は個々の標本データに対する評価値を平均や合計などで総合するものに適用可能である. 標本全体を用いずに, その一部を取り出した再標本を評価に用いる. 探索の開始時点では, 小さな再標本を用いて評価を行い, 探索が進むにつれて評価に用いる再標本を徐々に大きくして行く. 小さな再標本による評価は, 標本全体を用いるものに比べると大きな誤差を発生する確率が高く, 探索の初期段階では AIR は SA と同様にランダムな遷移を行う. 大きな再標本での評価は誤差が小さくなるので, 最終段階では, AIR は SA と同様に局所探索を行う. このように, AIR は SA と良く似た挙動を実現している. S-Map のピボット選択問題に AIR を用いれば, 1 遷移あたりの評価コストを格段に下げることができ, 十分な遷移回数を実行するよってヒューリスティックな解法より良い解を得ることが可能になった.

さらに, 本論文では, AIR が SA の近似になっていることを示す. そのために, SA と AIR がいずれも確率的揺らぎをもった目的関数を用いた山登りであるという統一的な視点でとらえることができることを理論的に示す. ベーカー関数 [2] を用いる SA の評価の揺らぎは logit 関数で与えることができる. AIR のそれは, 標準誤差, つまり正規分

布とみなせるので，probit 関数で与えることができる．Logit は probit で近似できるので，AIR は SA の近似として説明できる．

また，実験的には，SA は大域的最適解を求めるためには高温時に多くの試行が必要であることが知られている．これに対し，AIR では高温時には再標本サイズを小さくするため，結果の品質を損なうことなく，効率良く大域的最適解を求めることができる．このように評価用の標本サイズが大きいときには AIR が SA より速度的に非常に有利であることがわかる．

さらに，本論文では，AIR がどの程度まで SA の近似になるのかを検証するために，メトロポリス法によるモンテカルロシミュレーションにおいて logit 関数を probit 関数に置き換えた場合のシミュレーション誤差を測定した．これは，元々の SA がメトロポリス法を起源とするものであるので，AIR と SA の近似度合の検証となると考えたからである．その結果，状態遷移回数が 10^6 程度までは，シミュレーション誤差はほとんど生じず，10^7 を超えると徐々に誤差が大きくなることが確認できた．実際の AIR の適応状況においては，各温度帯に相当する範囲での状態遷移数はさほど多くないので，この誤差はほとんど問題とならないと思われる．

AIR が SA の近似であることを実証するための具体的問題として，S-Map のためのピボット群探索問題およびクラスタリング問題を取り上げる．S-Map のピボット群探索においては，遷移回数や温度と再標本数の対応などの条件をそろえた比較実験を行った結果，AIR と SA が非常に近い性能の解をみつけることを確認した．また，この時の速度面での AIR の大きな優位性についても確認した．さらに，有名な問題として *k*-means の対象となっていることでも知られるクラスタ重心との残差平方和の最小化問題を例に

あげ, その解法 SAGM[32] の SA 部分を AIR へと置き換えることによっても, 同程度の精度解が得られ, 速度面の優位性があることを示した.

1.5 論文の構成

本論文の第 2 章以降の構成は以下のようになっている. 第 2 章では, ヒルベルトソートの高性能化について議論する. 第 3 章では, 増加再標本焼きなまし法 AIR の提案と次元縮小射影 Simple-Map のピボット探索問題への適用結果について述べる. 第 4 章では, 焼きなまし法の共通視点を提示し, SA と AIR の近似性の理論的説明を与えるとともに, 実際の最適化問題における SA と AIR の比較実験を示す. 第 5 章では, 結論と今後の課題を示す.

第2章　ヒルベルトソートの高性能化

ヒルベルトソートは，ヒルベルト曲線に沿って高次元空間の点を並べ替える．ナイーブな実装では，最初にすべての点を分離するのに必要な解像度（次数）のヒルベルト曲線を描き，ヒルベルト曲線に沿った順番を表すヒルベルトインデックスと呼ばれる整数値を点に関連付け，最後に点とインデックスのペアをソートする．このような方法はヒルベルトインデックスを得たとしても，空間の次元数 m とヒルベルト曲線の次数 r の両方に関して指数関数的に大きなコストを必要とする．既知の改善された方法 [3, 15, 16] では，$O(mr)$ の時間で各点のヒルベルトインデックスを計算するため，n 個の点をソートするためには，インデックスの計算に $O(nmr)$ 時間，ソート全体では $O(nmr+n \log n)$ 時間を必要とする．

本章では，ヒルベルトインデックスを使わずにヒルベルト曲線に沿って n 個の点を $O(mnr)$ 時間で直接ソートするアルゴリズムを提案する．このアルゴリズムには，次の三つの利点がある．

- ヒルベルトインデックスのために余分なメモリ領域を必要としない．
- 同時に複数の点を扱うためヒルベルト曲線のシミュレーションコストが小さい．
- ヒルベルト曲線を不均一な解像度で，すなわち疎空間に対して低い次数で，密空間に対して高い次数でシミュレーションする．

提案アルゴリズムでは，ランダムデータにおいては全空間が疎になるため，$O(n \log n)$ 時間でソートすることができ，計算量を改善できる．また，マルチメディアデータの高次元の特徴など，実用的なデータにおいても，提案アルゴリズムを用いると非常に高速にソートすることが確認できる．

本章は，SISAP 2016 で発表した論文 [22] に基づいている．

2.1 高性能化の概要

提案アルゴリズムは，クイックソートと同様に二つのバケツへの振り分けを再帰的に繰り返すことによって並べ替えを行う．従来手法におけるヒルベルトインデックスを求める過程では，ヒルベルト曲線のシミュレーションを行っている．提案アルゴリズムでは，これと同等のシミュレーションを，ソート時の各ノードに対応して 1 ビット分ずつ進めていく．同じノードに属するデータ点は，そこに至るまでのビット列が一致しているため，複数のデータ点に対して一括でのシミュレーションが可能となる．新たに調べる 1 ビットによって，データ点はヒルベルト曲線順の前後に分かれるため，それぞれを二つのバケツへと分離し，新しいノードとして再帰的にソートを進める．バケツの中身が 1 個以下となったノードについては，それ以上のシミュレーションを必要としない．このことによって，過剰な精度を持つデータ点群が与えられた場合でも，無駄なシミュレーションを途中で打ち切ることができる．

従来のヒルベルトインデックスを求める手法では，ヒルベルトインデックスを求めるだけで 1 データ点あたり $O(mr)$ の計算量を要し，n 個のデータ点のヒ

ルベルトインデックス値を求めると $O(mnr)$ の計算量を要する．今回提案する手法では直接ソートを行うが，この計算量は一様乱数データの場合だと，$O(n \times \min(\log n, mr))$ となる．一般的に $\log n$ は mr より非常に小さいので，高速化につながる．実データでは一様乱数に対するほど高速化は期待できないが，最悪の場合でも従来と同じ $O(mnr)$ となる．今回実験に用いた実データでは，従来手法よりも速く動作することを確認した．

2.2 提案アルゴリズムの具体的手順例

ここで，提案手法の概要を具体例により説明する．提案アルゴリズムによるソート時の必要最低限のヒルベルト曲線シミュレーションの様子を一連の図によって示す．2 次元空間に図 2 のように 9 個の点が与えられているとする．これらをすべて区別するヒルベルト曲線は，次数 3 のものである（図 3）．ここで，図 3 においては, 空間が $2^{2\times3}$=64 個の小領域に分割されていることに注意しよう．図 3 の右側には 9 個の点の座標値を 10 進数と 2 進数で示している．

提案アルゴリズムは，空間を特定の次元で 2 等分することを繰り返す中，ヒルベルト曲線をシミュレートしながら，データをソートする．実際のアルゴリズムは再帰呼出による深さ優先で実行されるが，ここでの説明では幅優先探索を用いる．正方形や長方形に対するヒルベルト曲線の始点と終点，通過領域を有向矢印弧で描くことにする．図 4 では，領域全体の正方形を左下隅から右下隅へヒルベルト曲線を描くことを示している．

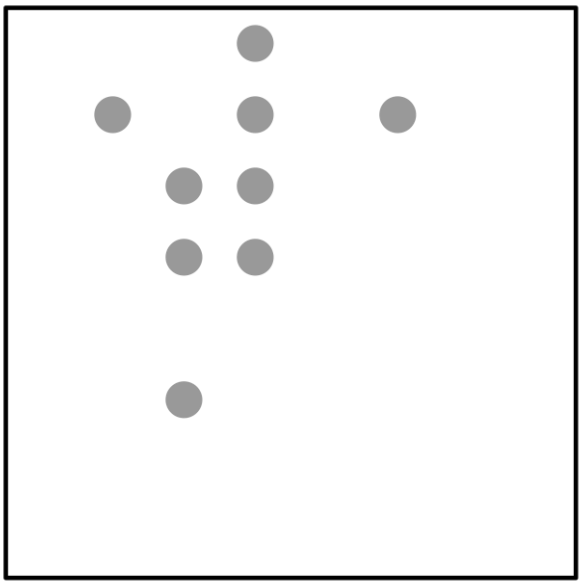


図 2　2 次元平面の 9 点

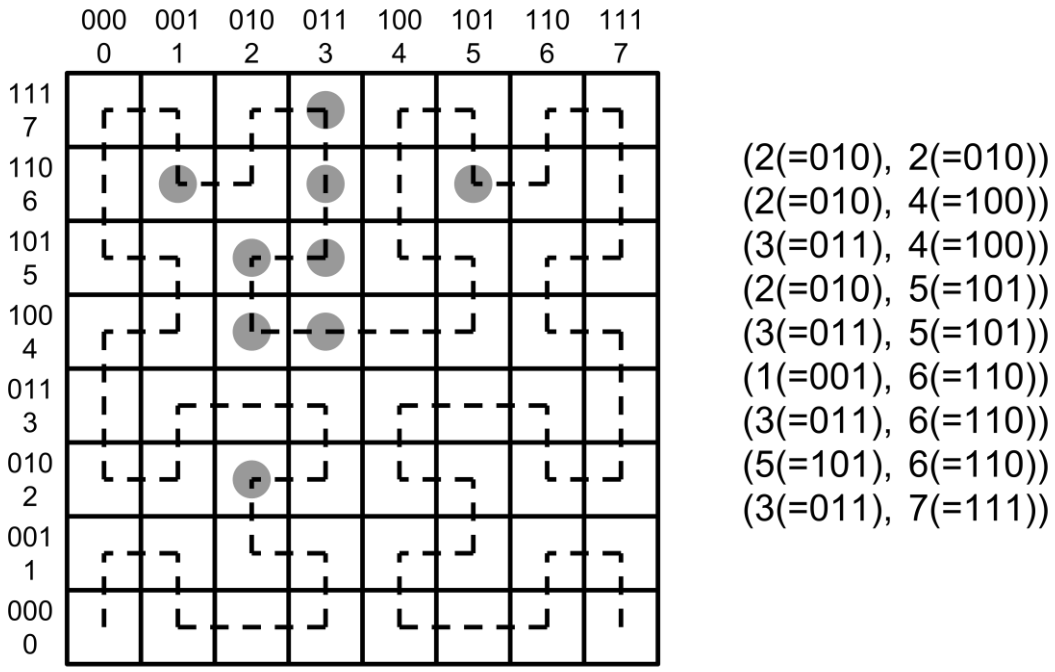


図 3　ヒルベルト曲線（2 次元，次数 3）

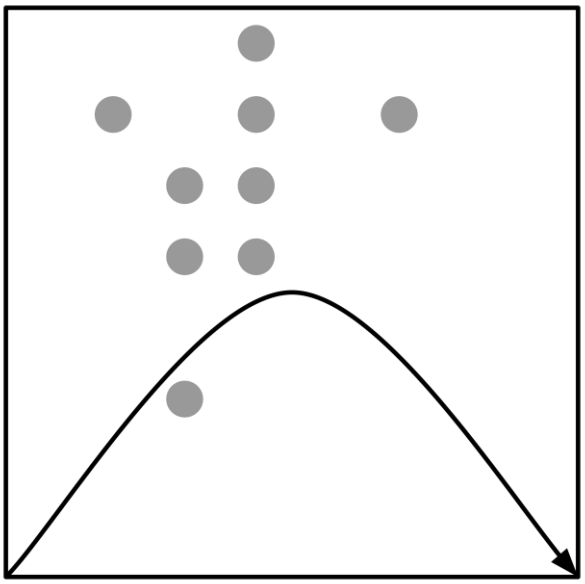


図 4　領域全体のヒルベルト曲線

まず，横軸で分割する（図 5）．分割で用いたビットは，図の右側の座標値を破線で囲んで示している．この段階で，左側の 9 個と右側 1 個の順序が決まっている．また，右側の領域には 1 点しか存在しないので，この領域のデータはソートが完了するので，これ以上の分割やシミュレーションは行わない．

つぎの縦軸での分割は，左側の領域のみで行う（図 6）．ここで，1 次のヒルベルト曲線のシミュレーションが終わっていることに注意しよう．

図 7 と図 8 に，それぞれ，2 次と 3 次のヒルベルト曲線のシミュレーションの様子を示す．図 8 の下図（一つ目の軸での分割）では，空のノードが発生していることが分かる．しかし，個数が 1 以下のノードへのシミュレーションはそこで打ち切られるため，無駄な処理は発生しない．

最終的にこの例では，9 個の点をソートするために，9 回の分割を行い，10 個の小領域に分けた．

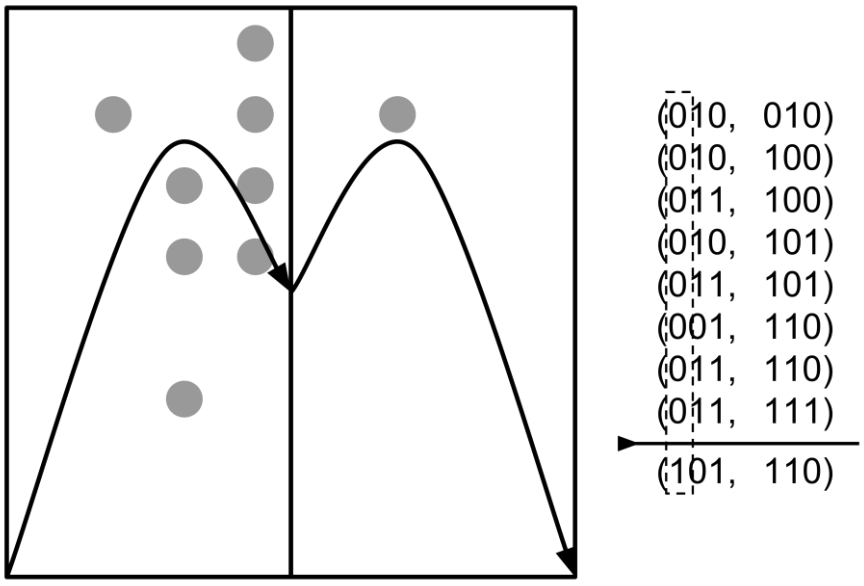


図 5　横軸での 2 分割

ヒルベルト曲線の最小単位シミュレーション回数は，ヒルベルトインデックスの算出の場合が *mnr*=54 回になるのに対して，ヒルベルトインデックスを算出しないヒルベルトソートでは，分割回数の 9 回と一致する．

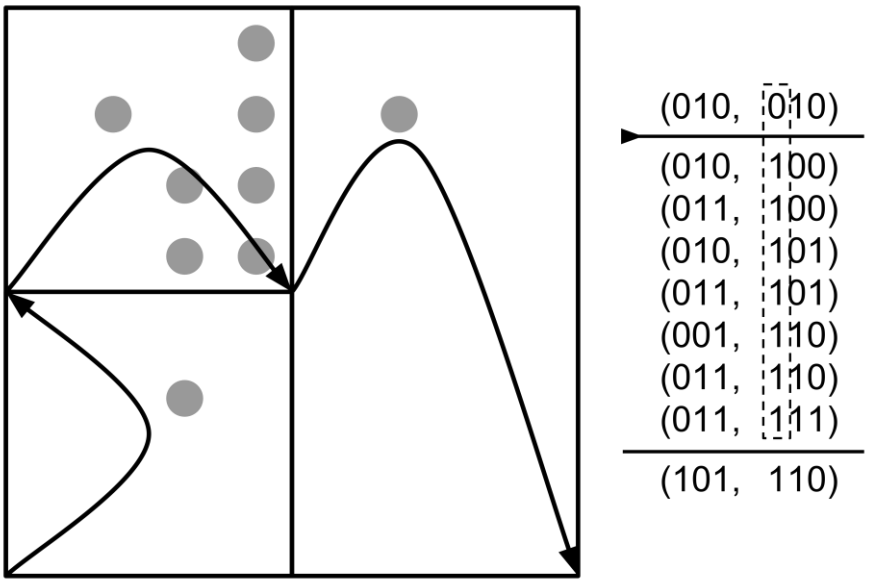
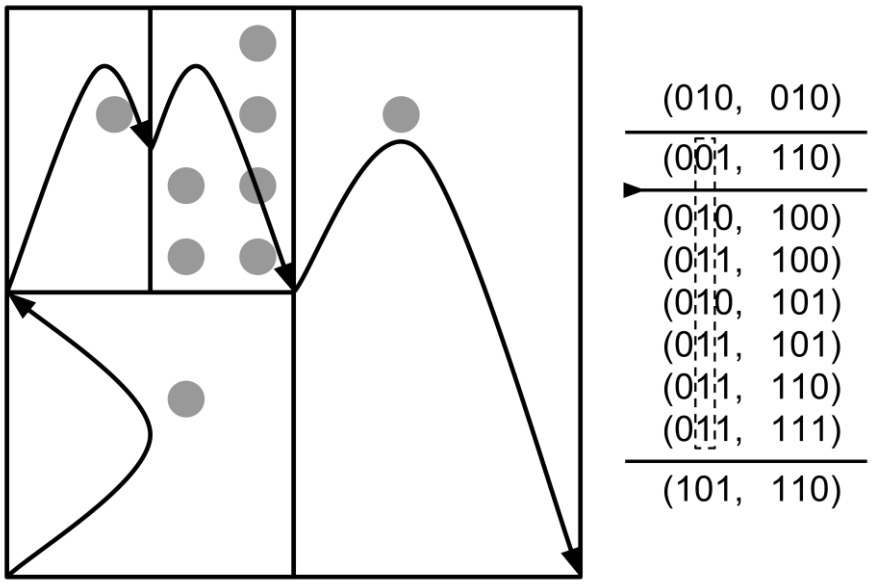


図 6　縦軸での分割



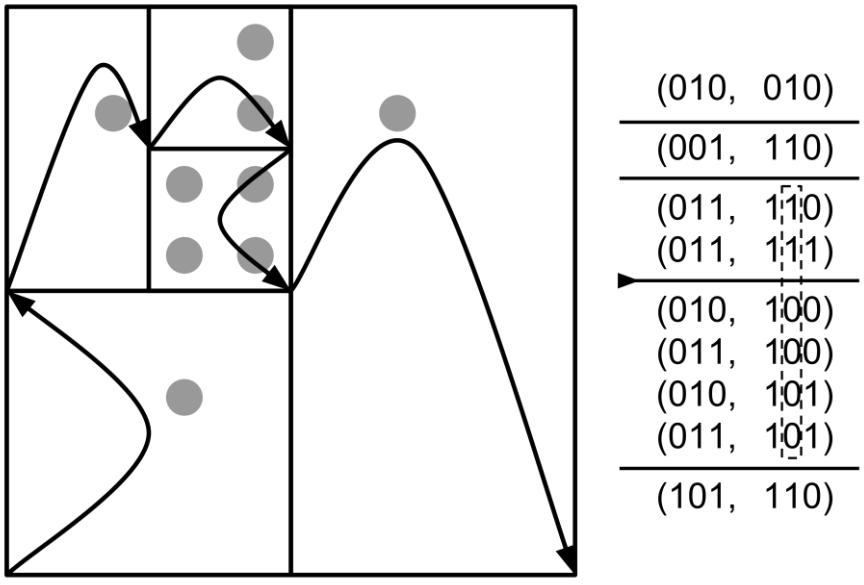
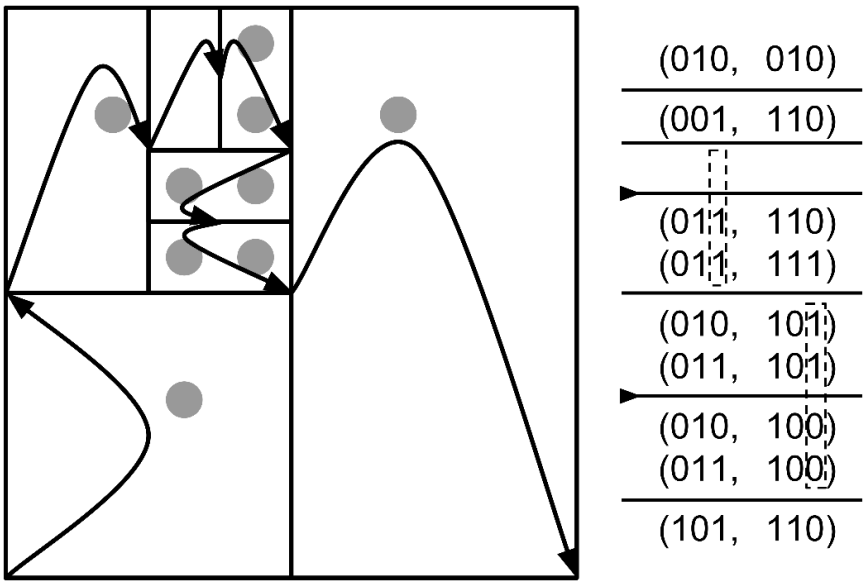


図 7　次数 2 のシミュレーションと分割



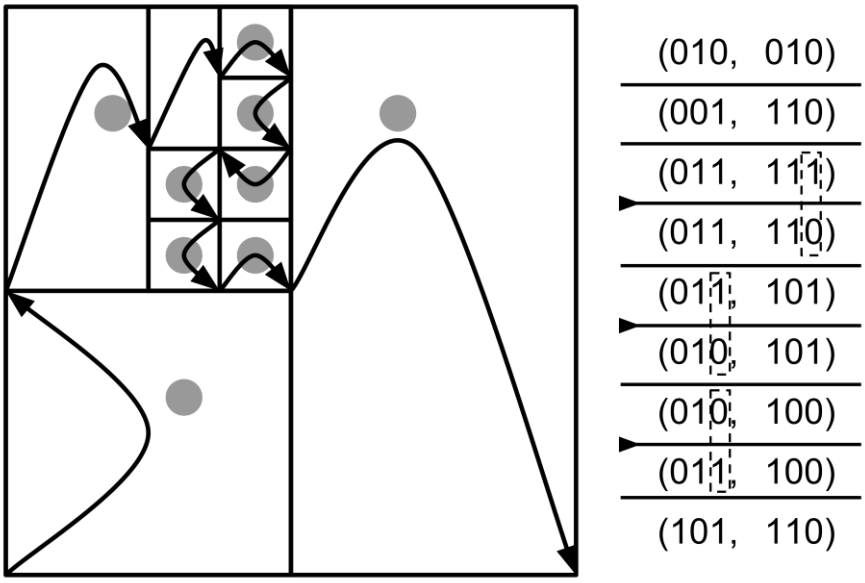


図 8　次数 3 のシミュレーションと分割

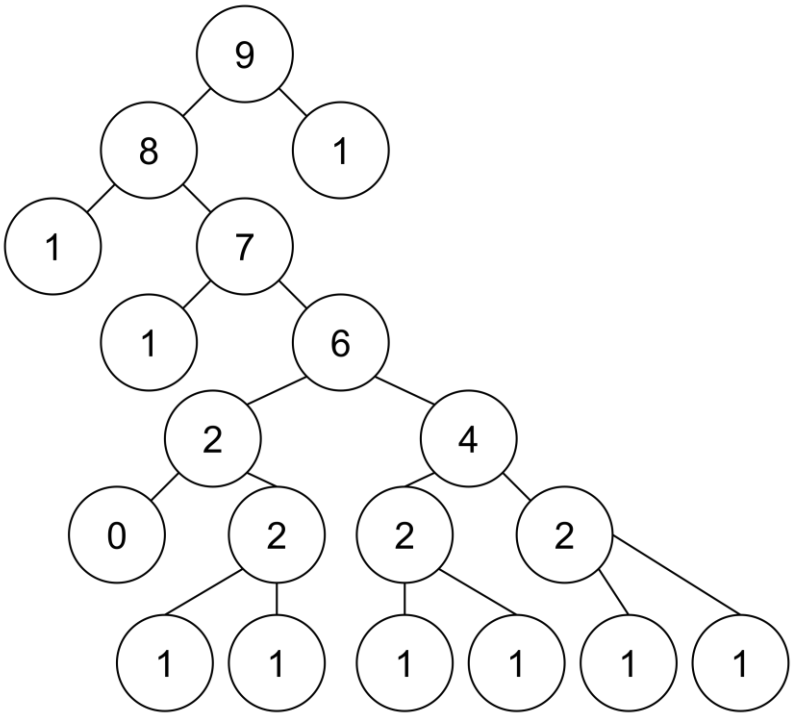


図 9　領域の分割と要素数

また，ヒルベルトインデックスの算出には $mnr = 54$ ビットの参照が行われるのに対して，ヒルベルトインデックスを算出しないヒルベルトソートでは，全ての中間ノードでの所属点の個数の総和に等しくなり，$9 + 8 + 7 + 6 + 2 + 4 = 42$ 回となる（図 9）．

この例では 6 割以上のデータ点が最終次数まで到達しているため劇的な改善が見られないが，たとえばこの配置のままで次数（座標の精度）を増やした時に，ヒルベルトインデックスの算出では次数に比例して参照ビット数も増えるが，ヒルベルトインデックスを算出しないヒルベルトソートでは参照ビット数は変化しない．実際にはこの差は，後の実験にも示すように，mr が大きいときには顕著になる．

2.3 提案アルゴリズム

図 10 と図 11 の疑似コードは，C++ 風に書いた．引数 A は便宜上，r ビット列の m 次元配列のポインタの n 個配列として疑似コードを書いたが，実際には $r = 8$ なら unsigned char，$r = 16$ なら unsigned short，$r = 32$ なら unsigned long，$r = 64$ なら unsigned long long などとなるかもしれず，その場合は flip や test を独自に実装すべきかもしれない．ここで，m は空間の次元数，r はヒルベルト曲線の次数，n はソートする点の個数である．

関数 HSort は，ヒルベルトソートを行うための関数である．戻り値はなく，引数 A へと副作用を及ぼすことによって目的を遂げる．

ここで，ソート対象は r-bit の符号なし整数を想定しているが，ここでは疑似コードの利便性上，bitset を用いている．Bitset の flip は，第一引数番目のビットを反転し，bitset の test は，第一引数番目のビットが立っているかどうかを返す（立っている場合は true，立っていない場合は false）．

関数 partition は，ヒルベルトソート内部で使われる，並べ替え関数である．引数 A へと副作用を及ぼすことによって並べ替えを実現し，さらに，ヒルベルト曲線順の前後の境界となった地点の添字を戻り値として返す．戻り値は，前者側を含まず，後者側を含む．この疑似コードにおいて，d 及び e は，各次数の開始時においては [15, 16] に出てくる d 及び e と同じ意味を持ち，c が 0 から $n-1$ までの 1 サイクルを通して，同様の変化を生じる．

関数 partition の計算量は，区間の長さと比例する．関数 HSort では，partition の呼出しと自身を再帰呼出しする以外は $O(1)$ の処理ばかりである．

```
void HSort(
   bitset<r> * A[],    // array of points to be sorted
   int st, int en,     // represent range A[st], ... , A[en]
   int od,             // current order of Hilbert curve
   int c,              // axis counter in current order
   bitset<r> & e,      // start positions of axes
   int d,              // axis number of first division on current order
   bool di,            // direction of previous division:
                       // false -> forward,      true -> backward
   int cnt             // number of continuous occurrences of di at same order
   )
{
  int p, d2;
  if(en <= st) return;     // nothing to do for empty or singleton
  p = partition(A, st, en, od, (d + c) % n, e.test((d + c) % m));
  if(c == m - 1) {         // simulation done, goto next order
    if(b == 0) return;     // no more order to simulate
    d2 = (d + m + m - (di ? 2 : cnt + 2)) % m;
    e.flip(d2); e.flip((d + c) % m);
    HSort(A, st, p - 1, b - 1, 0, e, d2, false, 0);
    e.flip((d + c) % m); e.flip(d2);    // undo of flips (2 before line)
    d2 = (d + m + m - (di ? cnt + 2 : 2)) % m;
    HSort(A, p, en, b - 1, 0, e, d2, false, 0);
  } else {
    HSort(A, st, p - 1, b, c + 1, e, d, false, di ? 1 : cnt + 1);
    e.flip((d + c) % m); e.flip((d + c + 1) % m);
    HSort(A, p, en, b, c + 1, e, d, true, di ? cnt + 1 : 1);
    e.flip((d + c + 1) % m); e.flip((d + c) % m);  // undo of flips
  }

}
```

図 10　関数 HSort

```
int partition(
    bitset<m> *A[], // array of points to be sorted
    int st, int en, // represent range A[st], ... , A[en]
    int od,         // order of interest
    int ax,         // axis number to be divided
    bool di         // direction, false -> ascending, true -> descending
    )
{
  int i, j;
  i = st - 1;
  j = en + 1;
  while(true) {
    do i = i + 1; while(A[i][ax].test(od) == di);
    do j = j - 1; while(A[j][ax].test(od) != di);
    if(j < i) return i;    // partition is completed
    swap(A[i], A[j]);
  }
}
```

図 11　関数 partition

ここで，HSort の 2 分割再帰呼び出しの区間の長さの和は増えることがないため，すべての階層の区間の長さの和は n 以下であることが保証される．また，区間が 1 以下になると打ち切りになるため，枝の数は各階層で n 以下であることが保証される．このことから，各階層の処理は枝分かれ分をすべて足したとして最悪でも $O(n)$ に収まることが分かる．途中打ち切りが発生しないとき，階層は全部で mr 階層となる．よって，全体の計算量は，$O(mrn)$ となることがわかる．

なお，乱数データでは，再帰呼び出しのたびに区間が半分ずつに分かれるこ

とが期待できるため，$O(\log n)$ 階層でソートが終了することを期待できる．よって，乱数データ時の期待計算量は $O(n \log n)$ となる．

2.4 ヒルベルトソートの速度比較実験

提案手法の特性を調べるため，速度比較実験を行った．ヒルベルトソートは，提案手法を著者が実装したものを用いた．ヒルベルトインデックスの算出は，[15, 16] における HILBERT INDEX 関数を著者が実装したものである．また，算出したヒルベルトインデックスを用いたソートとも比較を行った．実験では，C++STL の sort を用いた．

対象データとして，ランダムデータ，ランダムペアデータ，画像特徴データの 3 種類を用意した．

ランダムデータは，一様乱数によって調べるビットが常に半々程度に分かれることを想定したデータである．

ランダムペアデータは，一様乱数によって一つの点を生成した際に，もう一つ同じ点を生成する．この様にすることによって，提案手法は必ず最終次数までシミュレーションを行うことを強いられる．また，早々にノードは分かれるため，まとめてのシミュレーションの恩恵もあまり受けることができない．提案手法にとって，最も苦手と思われる状況を想定した．

画像特徴データは，動画から切り出した各コマ画像を，サムネイル化し，2 次元 FFT を用いて取り出した周波数特性特徴量である．700 万件以上のデータが存在し，そこからランダムサンプリングを用いて必要件数のデータを取り出し

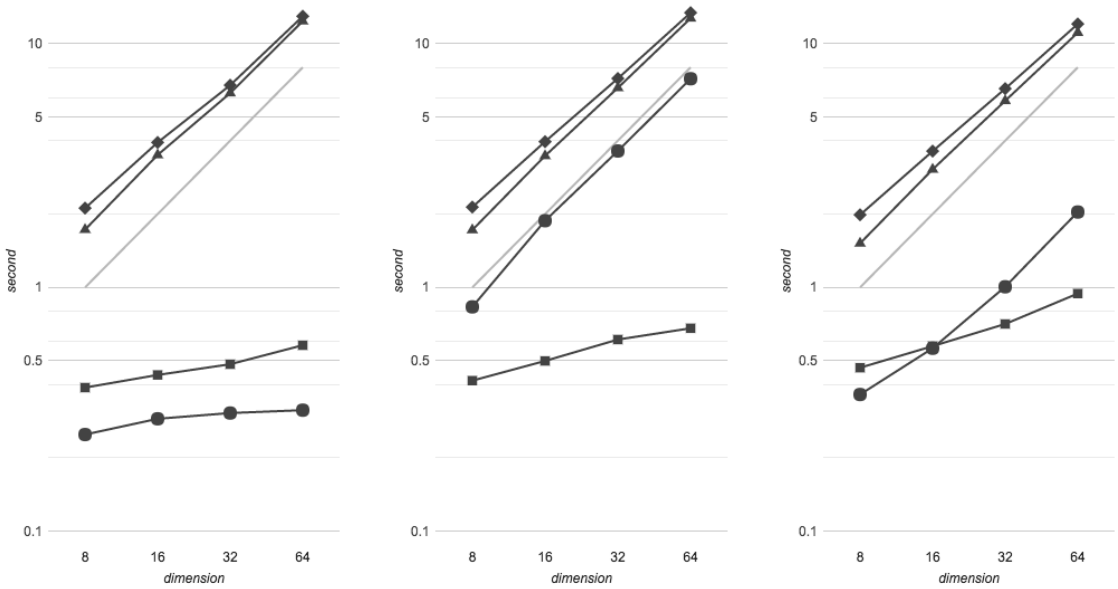
た.

画像特徴データの制約により, 次元数は最大で 64 次元とした. 次元数 m=64, 次数 r = 8 (=unsigned char に相当), データ件数 n = 1024k (= 1024×1024 = 1,048,576) を基準として, それぞれ次数とデータ件数を半分, 4 分の 1, 8 分の 1, 16 分の 1 へと減らした場合の比較を行った. 計算量を見るため, 縦軸横軸ともに対数目盛を用いた.

図 12 に, 次元数 m に対する計算時間の変化を示す. グラフでは, ●がヒルベルトソート, ▲がヒルベルトインデックスの算出, ■がヒルベルトインデックスのソート, ◆が▲と■の和を表す. ランダムデータでの次元数に対しては, ヒルベルトインデックスの算出計算量が線形なのに対して, ヒルベルトソートの計算量は線形よりはるかに小さく, 高速であることが分かった.

図 13 のランダムデータでのデータ件数 n に対する計算時間の変化においては, どの処理も線形であることが確認できる. 図 12 のランダムペアデータでの次元数の変化に対しては, 常に最終次数までのシミュレーションを強いられるために, ヒルベルトソートの計算量が線形になっていることが分かる. それでもなお, この実装では, 倍近く速いことが確認できている. 図 13 のランダムペアデータでのデータ件数変化においては, どの処理も線形である事が確認できる. 図 12 の画像特徴データでの次元数変化においては, ランダムペアデータと同様に, 計算量が線形になっていることが確認できる. 実際のデータでは, データに偏りがあるために, この様になるのだと思われる. この実装では, ヒルベルトソートの方が約 5 倍速く動作している.

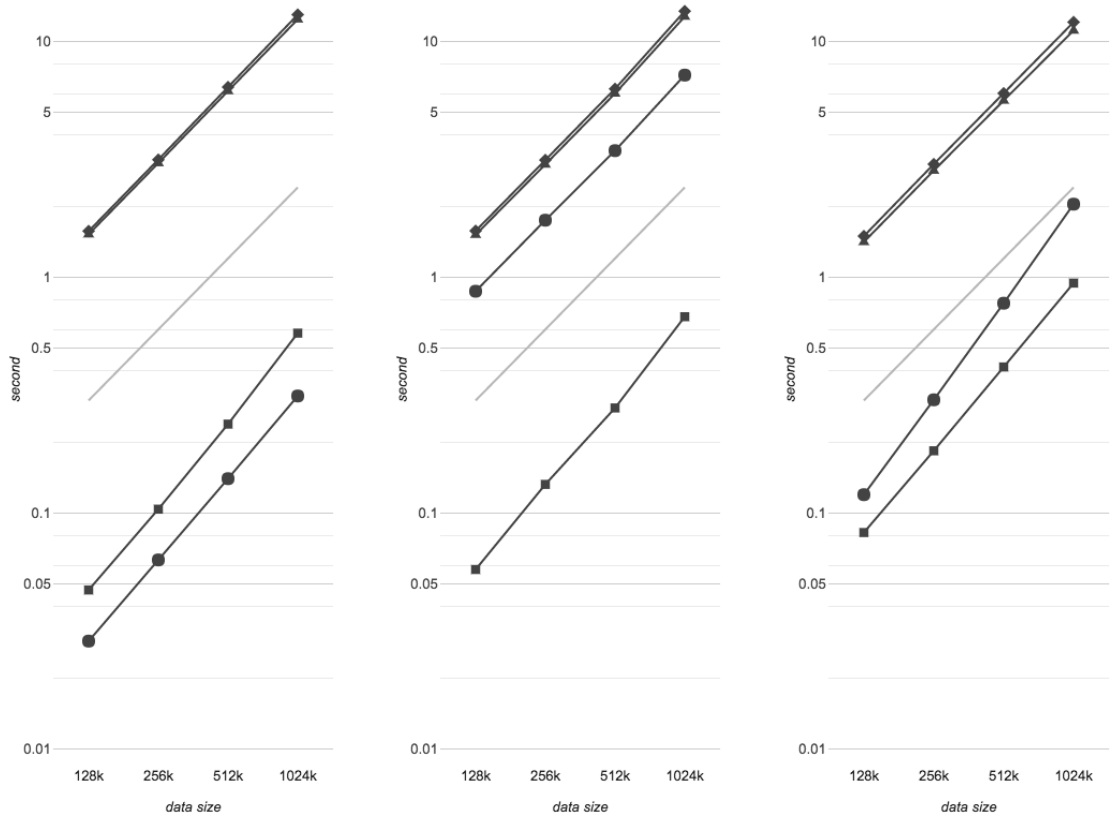
図 13 の画像特徴データでのデータ件数変化においては，ヒルベルトソートが線形よりも大きい傾きを示しているように見える．これはおそらく，偏りのあるデータのランダムサンプリングでは，データが増えるほどに同じ様なデータが重複しやすいためだと思われる．つまり，データ増加に伴って，ランダムデータの様な特性からランダムペアデータの様な特性へと，わずかながら近づいているものと思われる．



ランダムデータ　　ランダムペアデータ　　画像特徴データ

●：ヒルベルトソート，▲：ヒルベルトインデックスの算出，
■：ヒルベルトインデックスのソート，◆：▲＋■

図 12　次元数 m に対する計算時間の変化



ランダムデータ　　ランダムペアデータ　　画像特徴データ

●：ヒルベルトソート，▲：ヒルベルトインデックスの算出，
■：ヒルベルトインデックスのソート，◆：▲＋■

図 13　データ件数 n に対するソート時間の変化

2.5 結論

残念ながら，実用的な画像特徴データに対しては，計算時間のオーダーの改善は見られなかったが，十分なスピードアップを達成することができている．改良の理由の一つは，不均一な解像度のヒルベルト曲線をシミュレートするヒルベルトソートの利点によって説明することができる．可変長のインデックス

を導入すれば，ヒルベルトインデックスを計算するための同様の手法が得られる．しかし，新しいデータセットが追加されると，可変長インデックスを使用しても，古いデータのインデックスを再計算する必要がある．このように，ヒルベルトソートの手法は，動的な状況でより有利である．

実際，田島 [40] はヒルベルトインデックスを用いずに，ソート済みのデータを併合するアルゴリズムをヒルベルトマージとして実装している．このように，提案手法を用いることにより，Hilbert R-Tree[26] からヒルベルトインデックスを削除して省スペース化しても，効率良く構築・維持することが可能となる．

第3章　増加再標本焼きなまし法を用いた次元縮小射影 Simple-Map のピボット探索

本章では，Simple-Map やスケッチなどの次元縮小に最適なピボット集合を選択するために，増加再標本焼きなまし法（Annealing by Increasing Resampling, AIR）と名付けた手法を提案する．AIR では，すべての状態が標本を使用して評価されることを前提としている．AIR は，任意の初期状態から開始して，サイズが最初は小さく，徐々に増加するサイズの標本から無作為に選ばれる再標本を用いて状態評価しながら，山登りによって遷移を繰り返す．実験により，AIR が従来の方法よりも優れたピボット集合を焼きなまし法（SA）よりも短時間で発見可能なことが確認された．

本章は，LWDA 2017 で発表した論文 [23] に基づいている．

3.1 次元縮小射影 Simple-Map

ここでは，提案する最適化手法の適用対象である次元縮小射影 Simple-Map [38]（S-Map）を簡単に紹介する．S-Map は，Fréchet の埋め込み [31] の一種と見ることもできる．

二つの距離空間 $(U, D), (U', D')$ を考えよう．ここで，D および D' は，三角不等式を満たす距離関数とする．データuの次元数を$\dim(u)$とする．写像 π: $U \to U'$が**次元縮小**であるとは，任意の$u, v \in U$に対して，以下の条件を満たすことをいう．

$$\dim(\pi(u)) \leq \dim(u) \tag{1}$$

$$D'(\pi(u), \pi(v)) \leq D(u, v) \tag{2}$$

条件 (1) は，次元を縮小することを，条件 (2) は D' が D による距離の上限を与えることを意味する．

Simple-Map（S-Map）は，**ピボット**と呼ぶ点 p を用いて以下によって定義される射影 π_p に基づく．

$$\pi_p(u) = D(p, u)$$

三角不等式により，任意の $u, v \in U$ に対して，以下が成立つ．

$$\left|\pi_p(u) - \pi_p(v)\right| \leq D(u, v)$$

ピボット集合 $P = \{p_1, \ldots, p_{m'}\}$ を用いる S-Map π_P および D' を以下で定める．

$$\pi_P(u) = (\pi_{p_1}(u), \ldots, \pi_{p_{m'}}(u))$$

$$D'(\pi_P(u), \pi_P(v)) = \max_{i=1}^{m'} \left|\pi_{p_i}(u) - \pi_{p_i}(v)\right|$$

そうすると，m' が元の次元より小さくなっていれば，π_P は次元縮小になる．

S-Map では，距離が縮む．その縮み，つまり，距離の欠損は小さい方が類似検索に有用である．射影次元を大きくすると距離の縮みは小さくなるが，次元の呪いの影響を強く受ける．できるだけ低い次元でできるだけ距離の縮みを小さくすることが重要である．点対の集合 S に対する π_P の**距離保存率**とは，S

の点対間の距離の総和に対する，射影後の点対間の距離の総和の割合である．

3.2 増加再標本焼きなまし法

まず，ここで扱う最適化問題について，前提を述べておく．探索空間を A とし，その要素を**状態**と呼ぶ．**目的関数**（評価関数）$E(x,S)$ は，標本 S に対する状態 x の評価値を与えるもので，評価値は小さい（大きい）ほど好ましいとする．**最適化問題**とは，探索空間 A の中から評価値を最小（あるいは最大）にする x を求めることである．また，目的関数 E は，任意の部分標本 $S' \subseteq S$ と任意の状態 $x \in A$ に対して，

$$E(x,S') = \frac{1}{|S'|}\sum_{s\in S'} E(x,s)$$

と定義するものとする．ここで，$E(x,s)$ は標本中の個々のデータ $s \in S$ に対する任意の状態 $x \in A$ における評価とする．さらに，探索空間 A における x の**近傍** $Nb(x)$ が定められているとする．近傍 Nb については，焼きなまし法や局所探索法におけるものと同様に，

$$\forall x, y \in A, y \in Nb^*(x)$$

を満たしている，すなわち，任意の x から近傍操作 Nb を有限回適用すれば，任意の y が得られるとする．

図 14 に AIR のアルゴリズムを示す．ループの繰り返し回数 t を**時刻**と呼ぶことにする．ここで，$size: \mathbb{N} \longrightarrow \mathbb{N}$ は，増加関数で，時刻 t における標本全体 S に対する再標本のサイズを決める．$Trials$ は状態遷移の総数を表す．ここで，

任意の t に対して $size(t)$ が標本全体 S のサイズであるとき，AIR は，常に標本全体を用いた状態評価を用いるので，いわゆる局所探索と同等であることに注意しよう．現状態の近傍 $Nb(x)$ の中から再標本 S' に対する評価が良いものを探す手法の詳細については任意である．実際には，実装上や効率化のために，$Nb(x)$ の一部またはすべての中で，再標本 S' に対する評価が最も良いものに遷移する最急勾配法風にすることもある．

探索の初期段階では再標本サイズが小さいため, 再標本に対する評価 $E(x,S')$ と真の評価 $E(x,S)$ とのずれは大きくなる可能性が高く，真の評価が悪いものへ遷移する可能性がある．このように，$size(t)$ が小さいときには，AIR での遷移は, ランダムウォーク的になり, SA の高温のときと同様のものになる. 一方, $size(t)$ が標本全体のサイズに近くなると, $E(x,S') \fallingdotseq E(x,S)$ となり AIR は局所

function AIR(*S*: 標本):状態;

$x \leftarrow$ 任意の状態;

for t =1 **to** $Trials$

$x' \leftarrow Nb(x)$ から選んだ任意の状態;

$S' \leftarrow S$から無作為に選んだ再標本，ただし，$|S'| = size(t)$;

if $E(x,S') > E(x',S')$ **then**

$x \leftarrow x'$;

return x;

図 14　増加再標本焼きなまし法（AIR）

探索的な状態遷移を行うので，SA の低温のときと同様となる．

AIR の長所としては，探索の初期段階の探索が高速になることがある．なぜならば，再標本比 $size(t)$ が小さいときの状態評価が低コストで求められるからである．一方，通常の SA では，多数の標本を用いた場合には，高温状態での状態遷移回数を多くすることが困難である．

3.3 従来手法と提案手法の性能比較

実験に用いた多次元データは，約 1,700 本のビデオのコマ画像約 680 万件の特徴データで，次元数は 64 次元，各次元は 0～255 の整数，特徴間の距離は L_1 である．射影次元数（ピボット数）m' は，R-Tree の検索効率が良くなる 8 とした．評価用の標本 S は，データベースから任意に選んだ 5,000 個のデータ対を用いた．目標関数 E は，ピボット集合 P による S-Map 射影の部分標本 S' に対する**距離保存率**

$$E(P,S') = \frac{\sum_{(u,v)\in S'} D'(\pi_P(u),\pi_P(v))}{\sum_{(u,v)\in S'} D(u,v)}$$

を用いて，これを最大化するピボット集合 P を探索した．

局所探索と AIR で用いたピボット集合 P の近傍 $Nb(P)$ は，P から選んだピボット p に対して，p の 64 の軸から一つを 0～255 に変化させたものとした．

比較対象としては，過去に行った研究での SA や統計量に基づく 2 値量子化法 BQ，局所探索 LS に関する実験結果を示す．AIR では，状態遷移回数を SA や BQ と LS 同程度の計算時間となるように調整したものを表 1 に示した．

表 1 から，AIR は，SA と同程度の探索時間でより大きい距離保存率をもつピボットを求めることに成功していることがわかる．さらに，AIR は，BQ と比べると，より短い探索時間で同程度の距離保存率を達成している．また，LS は，探索時間を大きくするとより大きい距離保存率のピボットを見つけることができるが，AIR を用いると，それよりも短い探索時間で同程度のものを見つけることができている．このように，速度面，最適化の面，いずれにおいても，AIR が優れていることが分かる．

表 1　Simple-Map のピボット探索

最適化手法	探索時間（秒）	距離保存率
SA	43	55.93%
BQ	31	56.48%
LS	1386	57.32%
AIR	27	56.90%
	64	57.33%
	640	57.56%

3.4 提案手法の挙動

本章では，次元削減のための最適なピボット群を選択するために，再標本サイズを増やす手法 AIR を提案した． AIR は，表 1 に示すように，最適化に用いた評価関数の観点から，従来の方法よりもピボット群を効率的に見つけることができた．

このとき，探索途中の解のスコアの変化をグラフにしたものが図 15 である．AIR の探索途中では再標本を用いて解の評価を行っているが，この図のスコアは，途中で得られているピボット群を標本全体で評価した距離保存率を示している．赤と青で示した二つのグラフは，それぞれ総試行回数が 40k，400k のときのもので，表 1 における，それぞれ 64 秒，640 秒の結果に対応する．この探索途中におけるスコアの変化は，焼きなまし法（SA）を用いた場合と良く似ている．なぜ，再標本上で山登りを行う AIR が SA と近い挙動を示すのか，次章では，その部分に焦点を当てて解明を試みる．

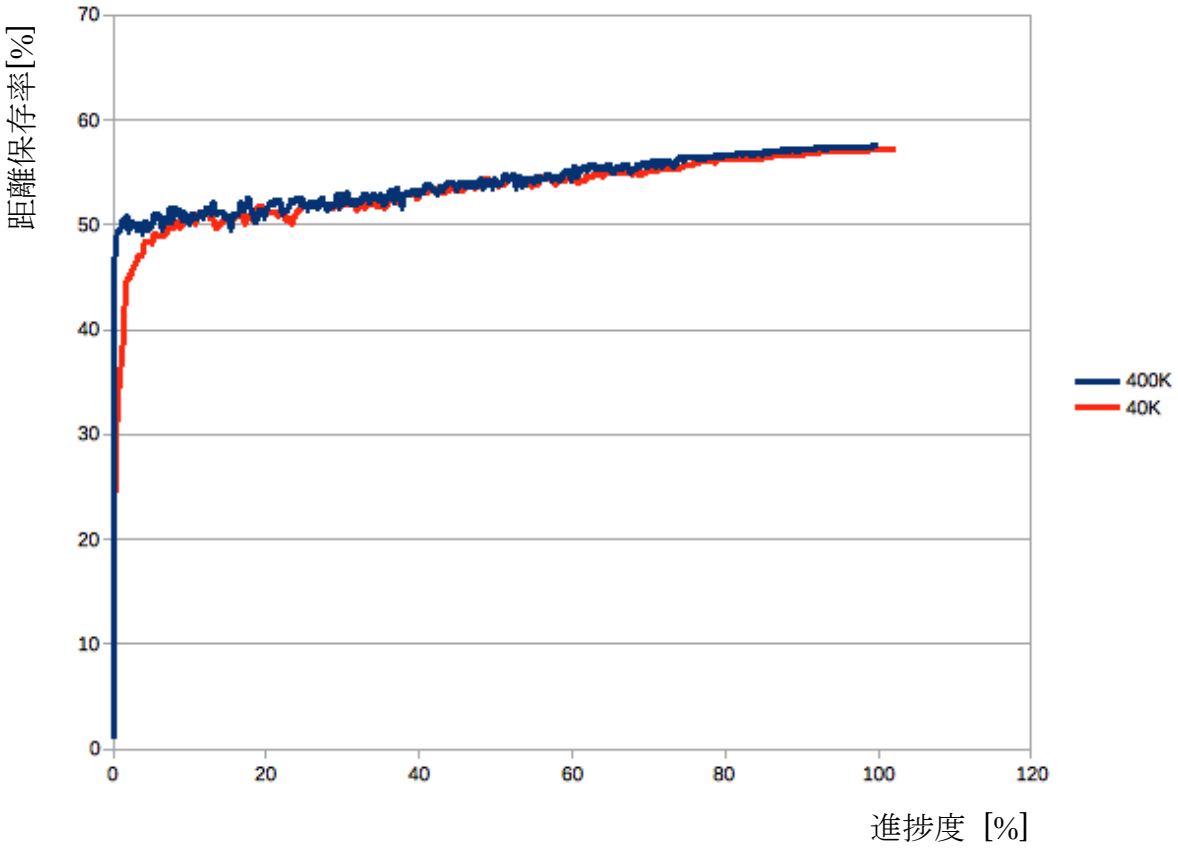


図 15　AIR における最適化の挙動

第4章　焼きなまし法の共通視点

増加再標本焼きなまし法（Annealing by Increasing Resampling, AIR）は，標本評価を用いる組合せ最適化問題に対して，少しずつ再標本サイズを増やしながら山登りを行う確率的な最適化法である．本章では，従来の焼きなまし法（Simulated Annealing, SA）と AIR の共通視点を導入する．この視点では，SA と AIR の両方を確率的な変動を伴う目的関数の確率的山登りと見ることができる．生じる確率的変動は，SA では logit，AIR では probit である．Logit と probit は近似の関係にあるため，AIR は SA の近似とみなすことができる．次元縮小射影のピボット探索や焼きなましによるクラスタリングなどの最適化問題に対する実験結果もまた，AIR が SA の近似であることを支持している．さらに，目的関数が多数の標本を必要とする場合，AIR は結果の品質を犠牲にすることなく，SA よりもはるかに高速に動作することも確認できる．

本章の内容は，ICPRAM 2019 で発表予定 [24] である．

4.1 焼きなまし法と提案手法の共通視点

ここで，焼きなまし法 SA[27] と増加再標本焼きなまし法 AIR[23] に共通の視点を与えておく．用いる記号や表記法を表 2 に示す．ここでは，標本に対する評価値を用いる目的（エネルギー）関数 $E: A \times 2^S \to \mathbb{R}$ の最小化問題を扱うことにする．ただし，A は探索空間，すなわち可能な解すべての集合，S は標本のデータ集合である．標本の部分集合を再標本という．$E(x, S)$ は，個々のデー

表 2　記号と記法

記法	意味
$t \in \mathbb{N}$	時刻（0, 1, 2, …）
$T(t) \geq 0$	時刻 t における温度（単調減少）
S	目的関数 E の評価用のデータ集合（標本）
$\|S\|$	集合 S の要素数
$size(t) \in \mathbb{N}$	時刻 t における再標本サイズ（$\leq \|S\|$）
A	探索空間
$x, x' \in A$	探索空間の要素，状態
$Nb(x) \subseteq A$	xの近傍
$E(x, S')$	xの再標本 $S' \subseteq S$に対する評価値 $E(x, S') = \frac{1}{\|S'\|} \sum_{s \in S'} E(x, s)$
$E(x, s)$	xのデータ$s \in S$ に対する評価値
ω	一様乱数（$0 < \omega < 1$）
$P(\Delta E, T(t))$	評価値差 ΔE 温度 $T(t)$ における受理確率

タ $s \in S$ に対する解 x の評価を合計や平均などによって集計するものとする．最適化の目標は，E を大域的に最小化する x^*，つまり，すべての $x \in A$ に対して，$E(x^*, S) \leq E(x, S)$ となる x^* を求めることである．

4.2 焼きなまし法

SA において，つぎの状態が悪くなっても受け入れることを許す確率を**受理確率** [1]（*acceptance probability*）あるいは**受理基準** [45]（*acceptance criterion*）という．アルゴリズム 1 に SA の概要を示す．2 状態 x と x' の標本 S に対する評価値の差を ΔE とする．

$$\Delta E = E(x', S) - E(x, S)$$

SA では，2 種類の受理確率が良く用いられる．一つは，メトロポリス（Metropolis[33]）関数 P_M で，最初に SA [27] で用いられたもので，標準的に用いられている．

$$P_M(\Delta E, T) = \min\{1, \exp(-\Delta E/T)\}$$

もう一つは，ベーカー（Baker[2]）関数 P_B （または，heat bath 関数 [1]）であり，ボルツマンマシン [46] の文脈で導入されたヘイスティングス

procedure SA

$x \leftarrow$ 任意の初期状態;

for $t = 1$ **to** ∞ **do**

$x' \leftarrow Nb(x)$ から選んだ任意の状態;

$\Delta E \leftarrow E(x') - E(x)$;

$\omega \leftarrow \mathrm{rand}(0, 1)$;

if $\omega \leq P(\Delta E, T(t))$ **then** $x \leftarrow x'$;

アルゴリズム 1 焼きなまし法（SA）

（Hastings[17]）関数の特別な場合である．

$$P_B(\Delta E, T) = \frac{1}{1 + \exp(\Delta E/T)}$$

さて，状態 x に続いて $x' \in Nb(x)$ が選ばれる条件を考えよう．メトロポリス関数に対しては，$\omega \leq \exp(-\Delta E/T)$ であるので，つぎの条件が得られる．

$$\Delta E + T \cdot \log(\omega) \leq 0$$

ベーカー関数に対しては，

$$\exp(\Delta E/T) \leq \frac{1-\omega}{\omega}$$

となるので，つぎの条件が得られる．

$$\Delta E + T \cdot \mathrm{logit}(\omega) \leq 0 \tag{1}$$

ここで，logit はつぎで定義されるものである．

$$\mathrm{logit}(\omega) = \log\frac{\omega}{1-\omega}$$

単純な山登り法においては，ΔE が 0 以下であれば，より良い状態 x' に遷移する．受理条件 (1) の左辺は，温度 T に比例する確率的揺らぎを伴った ΔE と見ることができる．

4.3 増加再標本焼きなまし法

AIR においては，目的関数は，標本 S から選んだ再標本 S' を用いるものとし，個々の評価値の平均を最小化する問題を扱う．S' のサイズが小さいほど評価値は大きく揺らぐという性質をもつ．AIR はこのことを利用した探索手法である．アルゴリズム 2 に AIR の概要を示す．

procedure AIR

$x \leftarrow$ 任意の初期状態;

for $t = 1$ **to** ∞ **do**

$x' \leftarrow Nb(x)$ から選んだ任意の状態;

$S' \leftarrow S$からランダムに選んだ再標本,

ただし，$|S'| = size(t)$ とする;

if $E(x', S') - E(x, S') \leq 0$ **then** $x \leftarrow x'$;

アルゴリズム 2 増加再標本焼きなまし法（AIR）

全標本サイズを $|S|$ とし，2 状態 x と x' の再標本 S' に対する評価値の差 $E(x', S') - E(x, S')$ は標準偏差 σ の正規分布に従うと仮定する．目的関数は，標本の個々のデータに対する評価値の平均や合計で求められるので，この仮定は中心極限定理から合理的である．

そうすると，サイズ $|S'|$ の再標本 S' に対する x と x' の評価値の差は，標準誤差を持った正規分布にしたがう．つまり，$E(x', S') - E(x, S')$ の値は，真の評価差 $\Delta E = E(x', S) - E(x, S)$ に標準偏差 $\frac{\sigma}{\sqrt{|S'|}} \cdot \sqrt{\frac{|S|-|S'|}{|S|-1}}$ の揺らぎを持ったものである．よって，ω を 0 から 1 の範囲の一様乱数とすると，AIR における受理条件は，つぎで与えられる．

$$\Delta E + \frac{\sigma}{\sqrt{|S'|}} \cdot \sqrt{\frac{|S|-|S'|}{|S|-1}} \cdot \mathrm{probit}(\omega) \leq 0 \qquad (2)$$

ここで，probit は標準正規分布の累積確率密度関数の逆関数である．ω を 0

から 1 の範囲の一様乱数とすると，probit(ω) は正規分布に従うことに注意されたい．

AIR では標本 S の再標本 S'を選択している．次の反復時には S'とは独立に再標本を選ぶ必要がある．良く似た方法として，再標本 S' を保持して，差分のみの少数のデータを S' に付加して用いることが考えられる．もとの S' に対する評価のための計算の途中結果を再利用できるので，評価を高速化するメリットがある．しかし，この方法は AIR では安易に用いてはならない．各時点での次状態の選択に関する確率的試行は独立になるようにしなくてはならない．このように本来，AIR では再標本を試行毎に独立に選ぶ必要があるが，処理の効率化のために，現在の再標本を再利用しても良いが，時々は独立な選択により再標本を取り換えなくてはならない．

4.4 焼きなまし法の共通視点

次状態候補の受理条件は，ベーカー関数に基づいた SA では式 (1) であり，AIR では式 (2) である．

$$\Delta E + T \cdot \mathrm{logit}(\omega) \leq 0 \tag{1}$$

$$\Delta E + \frac{\sigma}{\sqrt{|S'|}} \cdot \sqrt{\frac{|S|-|S'|}{|S|-1}} \cdot \mathrm{probit}(\omega) \leq 0 \tag{2}$$

どちらも，真の評価値の差 ΔE と確率的揺らぎをもった項との和である．それらの揺らぎは，それぞれ，logit と probit で与えられる．正規分布はロジスティック分布で近似できることが知られている．つまり，σ_0=1.65 のとき，図 16

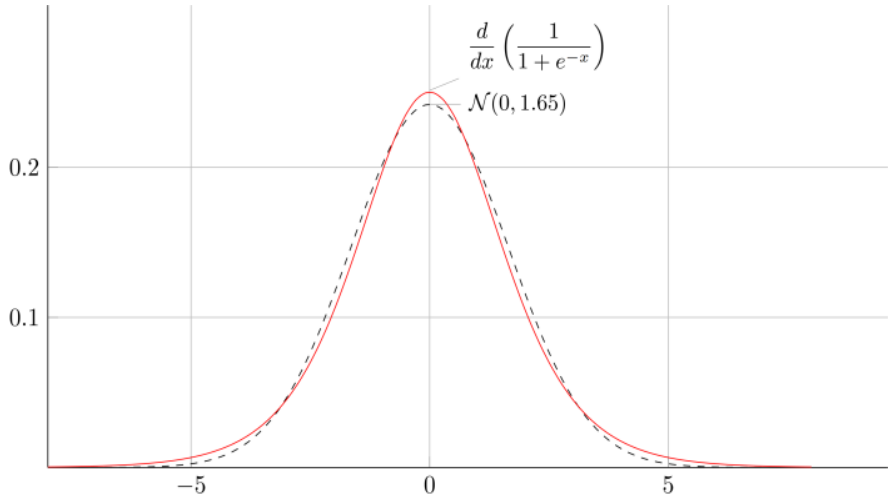


図 16　logit と probit

に示すように

$$\text{logit}(\omega) \approx \sigma_0 \cdot \text{probit}(\omega)$$

である[7].

したがって，SA と AIR の次状態候補の受理条件を一般化して，

$$\Delta E + \alpha(t) \cdot \Phi^{-1}(\omega) \leq 0$$

とすることができる．ここで，α は時刻 t に関して単調減少であり（AIR では，再標本サイズ n は，関数 $size$ で与えられることに注意），Φ^{-1} は，確率分布の累積密度関数の逆関数である．アルゴリズム 3 は，SA と AIR に対する共通視点を与える一般化アルゴリズムである．

4.5 焼きなましスケジュール

温度を下げていくスケジュールは，SA にとって，効率と精度の両面から重要である．ここで，SA で良く用いられている指数的冷却スケジュールに対応する

procedure Unified_Annealing

$x \leftarrow$ 任意の初期状態;

for $t = 1$ **to** ∞ **do**

$x' \leftarrow Nb(x)$ から選んだ任意の状態;

$\omega \leftarrow \mathrm{rand}(0, 1)$;

if $E(x') - E(x) + \alpha(t) \cdot \Phi^{-1}(\omega) \leq 0$ **then**

$x \leftarrow x'$;

アルゴリズム 3　SA と AIR の共通視点

AIR の再標本サイズのスケジュールの関係について述べておこう．温度スケジュールは，つぎで与えられるとする．

$$T = T_0 \cdot T_r \qquad (0 < T_r < 1)$$

ここで，T および T_0，T_r は，それぞれ，現在の温度，初期温度，温度比である．T_r は，時刻に対して単調減少であることに注意されたい．$|S|$ および n_0，$|S'|$ を，それぞれ，最大再標本サイズ（＝標本サイズ），初期再標本サイズ，現在の再標本サイズとする．また，σ_0をおおよそ 1.65 とし，σ を個々のデータの評価値の標準偏差とする．

次状態候補の受理条件は，ベーカー関数に基づいた SA では式 (1) であり，AIR では式 (2) である．どちらも，真の評価値の差 ΔE と確率的揺らぎをもった項との和である．それらの揺らぎが等しくなるようにするには，

$$T \cdot \sigma_0 = \frac{\sigma}{\sqrt{|S'|}} \cdot \sqrt{\frac{|S| - |S'|}{|S| - 1}}$$

とすれば良いことがわかる．これより，

$$|S'| = \frac{|S|}{(|S| - 1){T_0}^2 {T_r}^2 \frac{{\sigma_0}^2}{\sigma^2} + 1}$$

さらに，$T = T_0$ のとき，$T_r = 1$，$|S'| = n_0$ であるので，

$${T_0}^2 \frac{{\sigma_0}^2}{\sigma^2} = \frac{|S| - n_0}{(|S| - 1)n_0}$$

よって，

$$size(t) = |S^{'}| = \frac{|S|}{\frac{|S| - n_0}{n_0} {T_r}^2 + 1} \tag{3}$$

時刻 t における再標本サイズ $|S'|$ は，t の関数によって与えられる．ここで，$size(0) = n_0$ であることに注意しよう．式 (3) によって，SA における $T = T_0 \cdot T_r$ の冷却スケジュールと AIR における再標本サイズのスケジュールの橋渡しができる．SA と AIR で同じ温度比 T_r を用いるようにすれば，公平な比較実験を行うことができる．

4.6 MCMC における logit と probit

AIR がどの程度，焼きなまし法の近似になっているのかについて，焼きなまし法の基盤となっている MCMC（メトロポリス・ヘイスティングス法）を用いた分布の推定実験を行った．離散的に求めた実際の分布との相関係数を精度とし，その補確率を誤差とした．

対象関数としては，二つの正規分布を混合したつぎの 1 次元関数を用いた．

$$E(x) = \frac{0.3e^{-(x-1)^2} + 0.7e^{-(x+2)^2}}{\sqrt{\pi}}$$

受理条件としては，以下の 6 種類を比較した．ただし MCMC において温度は常に T=1 で固定となる．

i. メトロポリス法:

$$\omega \leq \min\{1, \exp(-\Delta E/T)\}$$

ii. ベーカー（ヘイスティングス）法:

$$\omega \leq \frac{1}{1 + \exp(\Delta E/T)}$$

iii. $\Phi^{-1}(\omega) = \log(\omega)$:

$$\Delta E + T \cdot \log(\omega) \leq 0$$

iv. $\Phi^{-1}(\omega) = \mathrm{logit}(\omega)$:

$$\Delta E + T \cdot \mathrm{logit}(\omega) \leq 0$$

v. $\Phi^{-1}(\omega) = 1.60 \cdot \mathrm{probit}(\omega)$:

$$\Delta E + T \cdot 1.60 \cdot \mathrm{probit}(\omega) \leq 0$$

vi. $\Phi^{-1}(\omega) = 1.65 \cdot \mathrm{probit}(\omega)$:

$$\Delta E + T \cdot 1.65 \cdot \mathrm{probit}(\omega) \leq 0$$

状態遷移回数を 10^5 から 10^8 まで変化させて，MCMC による誤差比較を行った結果を図 17 に示す．メトロポリス法（i）と log（iii）およびベーカー法（ii）と logit（iv）については，第 2 章で示したように数学的に等しく，ここで見られる様な差異は有意な差では

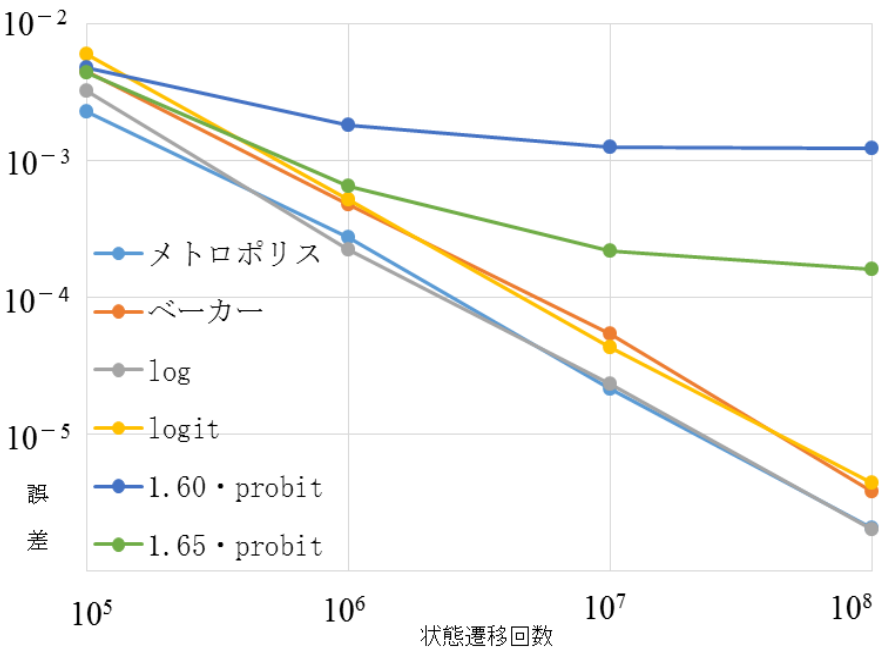


図 17　MCMC における誤差比較

ない．遷移回数 10^5 や 10^6 での実験では手法による誤差にはほとんど差が見られなかったが，10^7 や 10^8 での実験では有意な差が現れている．このことから，おおよそ遷移回数が 10^7 を超える付近から，logit と probit との差異が現れるのではないかと思われる．

この実験は，メトロポリス・ヘイスティング法の誤差に関する実験であるが，焼きなまし法の代替としてAIRを用いる場合には温度 T の存在が σ を吸収してくれるため，常に最適な σ_0 として機能し，厳密な最適値を求める手間が不要となる．また，焼きなまし法では温度 T を変化させるために，本実験で確認された 10^7 回という上限回数はおそらく，温度帯ごとの上限回数となるのではないかと考える．このため，焼きなまし法においては，より多い遷移回数にも耐えるものと考える．

4.7 次元縮小射影のピボット探索

本実験では，動画のコマ送り画像から抽出した約 700 万件 64 次元の特徴データを用いた．特徴間の類似度（相違度）は，L_1距離を用いた．次元縮小射影としては，Simple-Map[38]を用いた．Simple-Map の射影は，空間内の点をピボットとし，ピボットとデータ間の距離を射影像として用いる．Simple-Map は，三角不等式を満たす任意の距離空間に適用できるという特徴を持っている．ピボットを　個用いることで，m' 次元の L_∞ 距離空間に射影できる．2 点間の距離は射影後に縮むことはあっても伸びることはない．空間検索においては，ある程度低次元に射影して，できるだけ距離の縮みが小さくなるようにすることが重要である．本実験では，目的関数として距離保存率（射影前の距離の総和に対する射影後の距離の総和）を用いて，その最大化を行った．ピボット探索のための評価用の標本としては，$|S|$ =5,000 個のデータ点対をテストケースごとにデータベースから無作為に抽出したものを用いた．

実験環境としては，2.53GHz Intel Core i5，主記憶 8GB の mac OS X 環境上にて測定を行った．ここでは，SA と AIR とが近似になっていることを示すために，遷移回数を揃えた結果として報告する．遷移回数を 11×10^3 回，40×10^3 回，400×10^3 回でそれぞれそろえて行った．実験結果を表 3 に示す．

SA においても AIR においても，遷移回数を増やすことで同じぐらいのペースでスコア平均の上昇する様子が観測できた．処理速度については，SA より AIR の方が 5 倍から 8 倍程度高速になっており，その差は，遷移回数が増えるほど大きくなっていることが確認できる．SA と AIR の明らかな速度差は，AIR が高温時に全ての標本を評価せずに，その一部しか使わないことで処理を節約していることに起因する．今回の標

表 3　ピボット探索における SA と AIR の比較

手法	遷移回数（$\times 10^3$）	処理時間（秒）	距離保存率	
			平均	標準偏差
SA	11	149.7	57.06%	0.2763
	40	511.1	57.36%	0.2379
	400	4864	57.52%	0.1485
AIR	11	28.80	57.10%	0.2260
	40	70.49	57.35%	0.1333
	400	592.5	57.57%	0.1547

本サイズは $|S| = 5{,}000$ だったが，より大きな標本を用いることで，より大きな速度差を生むことが可能である．

4.8 焼きなましに基づくクラスタリング

SA の適用範囲として有名なナップサック問題や巡回セールスマン問題は，評価標本を用いる問題ではないため，AIR の適用の範囲外である．本実験では，SA を用いたクラスタリング手法に着目する．最小化する典型的な目的関数は，各点と最も近いクラスタ中心との二乗誤差の和（SSE）である．2013 年に Merendino と Celebi は，ガウス遷移を用いた中心摂動に基づく SA クラスタリングアルゴリズム SAGM を提案した[32]．SAGM は MMC と SMC の二つの冷却スケジュールを採用している．彼らは UCI 機械学習リポジトリ [8] の 10 個のデータセットを使用した実験を通して， SAGM（SMC）が他の SA アルゴリズムよりも大幅に速く収束することを報告した．表 4 に実験

表 4　データセット一覧

ID	データセット名	$\|S\|$	m	k
1	Ecoli	336	7	8
2	Glass	214	9	6
3	Ionosphere	351	34	2
4	Iris Bezdek	150	4	3
5	Landsat	6,435	36	6
6	Letter Recognition	20,000	16	26
7	Image Segmentation	2,310	19	7
8	Vehicle Silhouettes	846	18	4
9	Wine Quality	178	13	7
10	Yeast	1,484	8	10

で使用したデータセットの諸元を示した. ここで, $|S|$ はデータ数, m は属性数, k はクラス数である. データ数はさほど大きくないので, データセットをそのまま標本として用いた.

SAGM（MMC）と SAGM（SMC）を C++で実装し, それらの AIR 版についても互換スケジュールにて, それぞれ AIR（MMC）と AIR（SMC）として実装した. 実験環境は, Intel(R) Core(TM) i7-7820X CPU @ 3.60GHz, 主記憶 64GB の Windows10 環境上にて Windows Subsystem for Linux を用いて Ubuntu を動かし, その上での測定を行った. 初期ピボット（クラスタ中心）はデータベースから無作為に選んだものをテストケースとして, それぞれ 100 テストケースで実験を行った. 解の品質は SSE によって評価され, 値が小さいほど結果が良いことを示している. 表 6 で示す SSE 平均の比較より, SA と AIR

の間には品質面で大きな差がないことが確認できる. なお, 表 7 では, 良い方を太字で示している. また, 表 8 で示す処理時間平均では, 9 番のデータセットを除くすべてのデータセットで AIR は SA より高速に動作している. 9 番のデータセットでは標本サイズがとても小さい(|*S*|=178)ため, SA がわずかに速い. 実行時間に対する標本サイズ *N* の影響を観察するため, 図 18 に MMC および SMC における標本サイズに伴う SA と AIR の処理時間比を示している. 図から分かるように, 標本サイズが大きくなるほど AIR の方がより速くなり, 再標本による高速化の効果が現れていることが分かる.

表 9　解の品質(SSE 平均)

ID	MMC		SMC	
	SAGM	AIR	SAGM	AIR
1	17.55	**17.53**	17.60	**17.56**
2	**18.91**	19.05	**18.98**	19.08
3	**630.9**	638.8	**630.9**	646.8
4	6.988	**6.986**	**6.988**	6.991
5	1742	1742	1742	1742
6	2732	**2720**	2738	**2722**
7	411.9	**395.2**	413.1	**396.2**
8	225.7	**224.6**	225.8	**224.6**
9	37.83	**37.81**	37.85	**37.82**
10	**58.90**	59.08	**58.90**	59.04

表 10　処理時間平均(sec)

ID	MMC		SMC	
	SAGM	AIR	SAGM	AIR
1	1.727	1.376	0.282	0.182
2	0.719	0.629	0.125	0.091
3	1.549	0.857	0.281	0.095
4	0.139	0.110	0.022	0.015
5	17.40	1.051	3.216	0.125
6	167.4	13.72	28.82	1.803
7	3.285	0.606	0.523	0.056
8	1.298	0.367	0.219	0.035
9	0.786	0.814	0.141	0.149
10	2.854	0.971	0.497	0.088

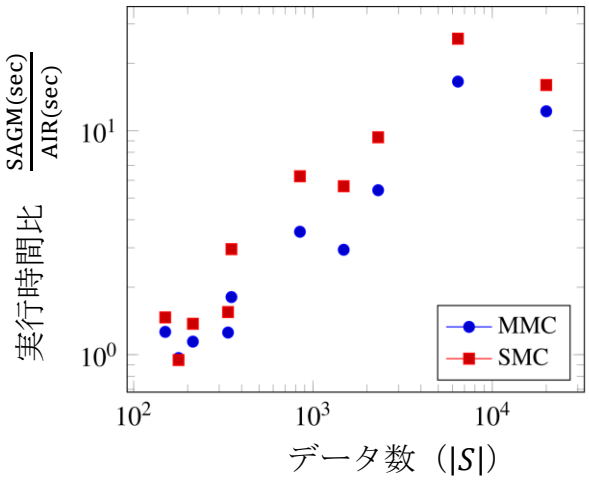


図 18　データ数と SA と AIR の実行時間比

4.9 結論

再標本評価による山登り法が焼きなまし法の近似となることを，logitとprobitの近似関係から理論的に示した．また，これについては，MCMC における誤差比較実験を行い，実際に同傾向の結果が得られることについても示した．

温度 T と関連の深い再標本サイズ $|S'|$ は，総標本数 $|S|$ に近付くまでは $|S|$ の影響をほとんど受けないため，全てのデータセットにおいて同程度の $|S'|$ を用いることが可能になる．このため，処理時間比較の実験において $|S|$ の大きなデータセットでは特に顕著な結果を得ることができた．

今回のクラスタリング問題の実験では比較対象の SA-Java（SAGM の Java 実装で M. Emre Celebi 教授より提供を受けたもの．独自実装の SAGM-C++版の動作確認に使用）があまりに重たかったために，最大の $|S|$ は 20,000 であり，より大きなデータ集合での実験を行えなかったが，ビッグデータが流行る昨今，$|S|$ が大きくなればなるほどに有利になる AIR は，大きなアドバンテージを有すると思う．これは，従来の SA では適用が難しかった分野への応用の可能性を示すものである．

第5章　結論と今後の課題

ヒルベルトソートの高性能化においては，実際のデータにおける計算量の改善までには至らなかったものの，動作のためのメモリを節約しつつ，充分に高速に動作する事が確認できた．また，ランダムデータにおいては計算量の改善に成功した．高次元かつランダムデータに近い特性を持つデータに対しては，圧倒的な高速化を期待できる．

増加再標本焼きなまし法 AIR の性能については，特にサイズの大きなデータに対する速度面において良い結果が得られている．ビッグデータが普及していく現代において，これは重要な意味を持っていると考える．

また，AIR についての理論的な考察を行った．再標本評価による山登り法が焼きなまし法の近似となることを，logit と probit の近似関係から示した．また，これについて実験を行い，実際に同傾向の結果が得られることについても示した．

さらに，温度 T と再標本サイズ $|S'|$ の互換性の関係について，理論上の式を示すことができた．これによって従来の SA で経験的に事前調査した初期温度 T_0 を定めていた代わりにAIR では経験的に事前調査した初期再標本サイズ n_0 を与えることで，SA と理論的に同程度となる温度スケジュールが適用可能になった．ただし，これについて ICPRAM 2019 へと投稿後に追加調査で分かった新事実がある．図 19 に S-Map のピボット探索において，実際に理論上温度スケジュールが同程度になると思われる AIR と SA のスコア途中経過の比較をグラフに示した．かなり近い動きを

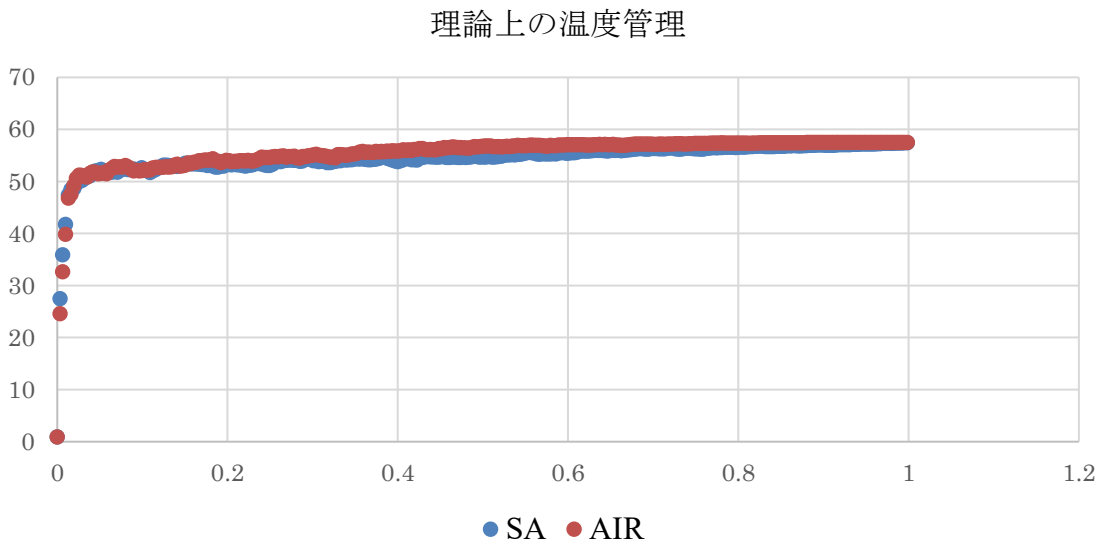


図 19　AIR と SA のスコア途中経過の比較

しているものの, 詳しく見ると AIR の方が若干早くスコア上昇が起こっており, AIR の方がやや早く温度が低下している様に思われる. この原因について, S-Map ではピボットの軸ごとに遷移を行っているため, 軸ごとに σ_0 が異なっているのかもしれない. しかし, 詳細については, 本論文をまとめるにあたって, まだ解明できていない. この現象の解明については, 今後行っていく必要がある.

温度 T と関連の深い再標本サイズ $|S'|$ は, 総標本サイズ $|S|$ に近付くまでは $|S|$ の影響をほとんど受けないため, 全てのデータセットにおいて同程度の $|S'|$ を用いることが可能になる. このため, 処理時間比較の実験において $|S|$ の大きなデータセットでは特に顕著な AIR の優位性を示す結果を得ることができた. 今回の焼きなましによるクラスタリングの対象問題の実験では, 比較対象である SA-Java の計算に時間がかかったために, 最大の $|S|$ は 20,000 であり, より大きな $|S|$ での実験を行えなかった. ビッグデータが流行る昨今, $|S|$ が大きくなればなるほどに有利になる AIR は, 大きなアドバンテージを有すると思われる. これは, 従来の SA では適用

が難しかった分野への応用の可能性を示すものである.

なお, 著者は, プログラミングコンテストにおけるハイパーパラメータのチューニングに AIR を用いている. あまり時間をかけずに大量のハイパーパラメータを適用してもほど良くチューニングしてくれるため, 使い勝手が良い. AIR は, 当初, 次元縮小射影 Simple-Map の最適化のために考案したものであるが, その背景にある直観は, 著者のプログラミングコンテストにおける経験に基づいていると思っている. 論文としてまとめていくことにより, AIR の意義について, より深く認識できたことは非常に有益であった.

本論文では, ヒルベルトソートと AIR の大きく分けて二つの課題について取り扱った. これらは一見して関係が薄そうに見えるかもしれないが, どちらも空間索引を支える上で重要な役割を担える可能性を著者は信じている. 著者が知る限りにおいては, 空間索引による最近傍検索の厳密解では未だ最悪の計算量が $O(|S|)$ より小さくなる報告はないが, 解を厳密解に限らなければ改善の余地について思いを巡らすことができる. たとえば, ヒルベルト曲線順のデータ列を二分探索して直前直後のデータ点を調べる操作は $O(\log|S|)$ で実行可能である. 高次元空間におけるヒルベルト曲線は複数の書き方が存在するので, それらに対応する複数のヒルベルトソート索引を用意しておけば, ヒルベルト曲線順での最近傍の列挙の中に実距離上での最近傍が含まれる確率をそれなりに確保できるかもしれない.

また, 論文をまとめるにあたって, AIR の適用範囲について色々と可能性を探った結果として, 元々の動機となった次元縮小射影のピボット探索以外に, *k*-means 法が対象としているクラスタリング問題を取り上げた. このクラスタリング問題もまた, 空間索引

と関連性が深く，AIR の適用対象として空間データベースが向いているのは確かであると考えている．今後，空間索引における新しい次元縮小射影が見つかった場合でも，その最適化問題を解くために AIR が活躍できる可能性は非常に高いと思う．

謝辞

本研究を遂行するにあたり，主査としてご指導頂きました平田耕一先生，学外から副査としてご指導頂いた学習院大学の久保山哲二先生，そして最終講義を終えながらも変わらずご指導を続けて頂いた副査の篠原武先生へと，心よりお礼申し上げます．また，副査として審査を通してご指導頂きました坂本比呂志先生と久代紀之先生にも大変お世話になり，感謝致しております．

また，第 4 章の実験において，SAGM とそのためのスケジュール SMC と MMC をソースコードとして提供してくださった M. Emre Celebi 教授に感謝します．

参考文献

1 S. Anily, A. Federgruen, “Simulated annealing methods with general acceptance probabilities,” J. App. Prob., 24: pp. 657–667, (1987).

2 A. A. Barker, “Monte Carlo calculations of the radial distribution functions for a proton-electron plasma,” Aust. J. Phys., 18: pp. 119–133, (1965).

3 B. Bustos, G. Navarro, E. Chavez, “Pivot selection techniques for proximity searching in metric spaces,” In Proc. Computer Science Society SCCC’01, XXI International Conference of the Chilean, pp. 33– 40. IEEE. (2001).

4 A. R. Butz, “Alternative algorithm for Hilbert’s space-filling curve,” IEEE Trans. on Computers, pp. 424–426, (1971).

5 E. Chavez, G. Navarro, R. Baeza-Yates, J. Marroqu, “Searching in metric spaces,” ACMComput. Surv. vol. 33, pp. 273–321, (2001).

6 P. Ciaccia, M. Patella, P. Zezula, “M-tree: An efficient access method for similarity search in metric spaces,” In Proc. VLDB’97, pp. 426–435, (1997).

7 E. Demidenko, “Mixed Models: Theory and Applications with R,” Wiley, 2nd edition. (2013).

8 D. Dheeru, E. Karra Taniskidou, “UCI machine learning repository.” University of California, Irvine, School of Information and Computer Sciences, http://archive.ics.uci.edu/ml. (2017).

9 W. Dong, M. Charikar, W. Dong, K. Li, “Asymmetric distance estimation with sketches for similarity search in high-dimensional spaces,” In Proc. ACM SIGIR’08, pp. 123–130, (2008).

10 F. K. Došilović, M. Brčić, N. Hlupić, “Explainable artificial intelligence: A survey,” In Proc. MIPRO 2018, 6 pages, (2018).

11 C. Faloutsos, K. I. Lin, “Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets,” In Proc. ACM SIGMOD’95, Vol. 24, pp. 163–174, (1995).

12 C. Faloutsos, S. Roseman, “Fractals for secondary key retrieval,” In Proc. 8th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pp. 247–252, (1989).

13 K. Fukunaga, “Statistical pattern recognition,” (second edition), (1990).

14 A. Guttman, “R-trees: A dynamic index structure for spatial searching,” Proc. SIGMOD’84, pp. 47–57 (1984).

15 C. Hamilton, “Compact Hilbert indices,” Dalhousie University, Faculty of Computer Science, Technical Report CS-2006-07, (2006).

16 C. Hamilton, “Compact Hilbert indices: Space-filling curves for domains with unequal side lengths,” Information Processing Letters 105, pp. 155–163, (2008).

17 W. K. Hastings, “Monte Carlo sampling methods using Markov chains and their applications,” Biometoroka, 57: pp. 97–109, (1970).

18 N. ハウ, 篠原武, 量子化を利用した次元縮小射影 Simple-Map の中心点探索に関する研究, 火の国情報シンポジウム, 情報処理学会九州支部, (2015).

19 N. Higuchi, Y. Imamura, T. Kuboyama, K. Hirata, T. Shinohara, “Nearest neighbor search using sketches as quantized images of dimension reduction,” In Proc. ICPRAM 2018, pp. 356–363, (2018).

20 N. Higuchi, Y. Imamura, T. Kuboyama, K. Hirata, T. Shinohara, “Fast nearest neighbor search with narrow 16-bit sketch,” to appear in Proc. ICPRAM 2019,

(2019).

21 D. Hilbert, “Über die stetige Abbildung einer Linie auf ein Flächenstück,” Math. Ann. 38, pp. 459–460, (1891).

22 Y. Imamura, T. Shinohara, K. Hirata, T. Kuboyama, “Fast Hilbert sort algorithm without using Hilbert indices,” In Proc. SISAP 2016, LNCS 9939, pp. 259–267, (2016).

23 Y. Imamura, N. Higuchi, T. Kuboyama, K. Hirata, T. Shinohara, “Pivot selection for dimension reduction using annealing by increasing resampling,” In Proc. Learn. Wissen. Daten. Analysen, LWDA’17, pp. 15–24. (2017).

24 Y. Imamura, N. Higuchi, T. Shinohara, K. Hirata, T. Kuboyama, “Annealing by increasing resampling in the unified view of simulated annealing,” to appear in Proc. ICPRAM 2019, (2019).

25 S. Kamata, A. Perez, E. Kawaguchi, “A computation of Hilbert’s curves in N dimensional space,” IEICE, J76-D-II, pp. 797–801, (1993).

26 I. Kamel, C. Faloutsos, “Hilbert R-tree: An Improved R-tree using fractals,” the 20th International Conference on Very Large Data Bases (VLDB), pp. 500–509, (1994).

27 S. Kirkpatrick, C. D. Gelatt Jr., “Optimization by simulated annealing,” Science, 220: pp. 671–680, (1983).

28 A. Krizhevsky, I. Sutskever, G.E. Hinton, “ImageNet classification with deep convolutional neural networks,” Advances in Neural Information Processing Systems 25, pp. 1097–1105, (2012).

29 R. Mao, W. Miranker, D.P. Miranker, “Pivot Selection: dimension reduction for distance-based indexing,” J. Discret. Algo. vol. 13, 32–46 (2012).

30 R. Mao, P. Zhang, X. Li, X. Liu, M. Lu, M, “Pivot selection for metric-space

indexing, In-ternat," J. Mach. Learn. Cybernet. vol. 7, pp. 311–323 (2016).

31 J. Matousek, Lectures on discrete geometry, Springer Verlag, (2002).

32 S. Merendino, M. E. Celebi, "A simulated annealing clustering algorithm based on center perturbation using Gaussian mutation," In Proc. FLAIRS Conference, pp. 456–461, (2013).

33 N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, "Equation of state calculations by fast computing machines," J. Chem. Phys., 21: pp. 1086– 1092, (1953).

34 V. Mic, D. Novak, P. Zezula, "Speeding up similarity search by sketches," In Proc. SISAP 2016, pp. 250–258, (2016).

35 A. J. Müller-Molina, T. Shinohara, "Efficient similarity search by reducing i/o with compressed sketches," In Proc. SISAP 2009, pp. 30–38, (2009).

36 大野真吾, 多次元データベースの近似検索のための量子化次元縮小射影に関する研究, 九州工業大学大学院修士論文, (2012).

37 T. Shinohara, J. Chen, H. Ishizaka, "H-map: A dimension reduction mapping for approximate retrieval of multi-dimensional data," In Proc. DS'99, LNAI 1721, pp. 299–305, (1999).

38 T. Shinohara, H. Ishizaka, "On dimension reduction mappings for approximate retrieval of multi-dimensional data," Progress in Discovery Science, LNCS 2281, pp. 89–94, (2002).

39 田中晶, 空間索引のための多次元データ順序付けの高速化に関する研究, 九州工業大学大学院修士論文, (2001).

40 田島圭, ヒルベルト曲線順序付けを用いた空間索引構造に対する効率的な挿入

法に関する研究, 九州工業大学大学院修士論文, (2011).

41 R. Wagner, M. Fischer, “The string-to-string correction problem,” J. ACM, Vol. 21, pp. 168–178, (1974).

42 Z. Wang, W. Dong, W. Josephson, M. Charikar Q. Lv, K. Li, “Sizing sketches: A rank-based analysis for similarity search,” In Proc. ACM SIGMETRICS’07, pp. 157–168, (2007).

43 P. Zezula, P. Savino, G. Amato, F. Rabitti, “Approximate similarity retrieval with m-trees,” VLDB J. vol. 7, pp. 275–293, (1998).

44 P. Zezula, G. Amato, V. Dohnal, M. Batko, “Similarity Search - The Metric Space Approach,” Advances in Database Systems 32, Kluwer (2006).

45 Schuur, P. C. (1997). Classification of acceptance criteria for the simulated annealing algorithm, Math. Oper. Res., 22, pp. 266–275.

46 Aarts, E. and Korst, J. (1989). Simulated annealing and Boltzmann machines: A stochastic approach to combinatorial optimization and neural computing, Wiley.